first.

Calculate

If no paths have been calculated, Calculate Paths will create a new motion path in cache based on the options specified in the pop-up menu or the Adjust Last Operation panel. Note, if the current context is an Armature calculating the objects motion paths, and not the bones, this operator will calculate the motion paths for all the bones within the armature as well.

Start, End

These are the start/end frames of the range in which motion paths are shown. The start frame is *inclusive*, so if you set *Start* to 1, you will really see the frame 1 as starting point of the paths…

Bake Location

Which point on the bones is used when calculating paths. Only available for bones while in Pose Mode.

Heads::	Calculates the path position of the bone's heads.
Tails::	Calculates the path position of the bone's heads.

Update Paths

In the case a path has already been calculated, this operator will update the path shape to the current animation. To change the frame range of the calculated path, you need to delete the path and calculate it again.

(Clear Paths)

Clears paths on all objects/bones or just the selected ones when holding `Shift`.

Update All Paths

Recalculates the motion paths for all visible objects and poses.

Display

Frame Numbers

When enabled, a small number appears next to each frame dot on the path, which is of course the number of the corresponding frame.

Keyframes

When enabled, big yellow square dots are displayed on motion paths, showing the keyframes of their bones (i.e. only the paths of keyed bones at a given frame get a yellow dot at this frame).

+ Non-Grouped Keyframes

For bone motion paths, it searches the whole Action for keyframes instead of in groups with matching name only (this is slower).

Keyframe Numbers

When enabled, you will see the numbers of the displayed keyframes, so this option is obviously only valid when *Show Keys* is enabled.

Lines

Toggles whether the lines between the points are shown.

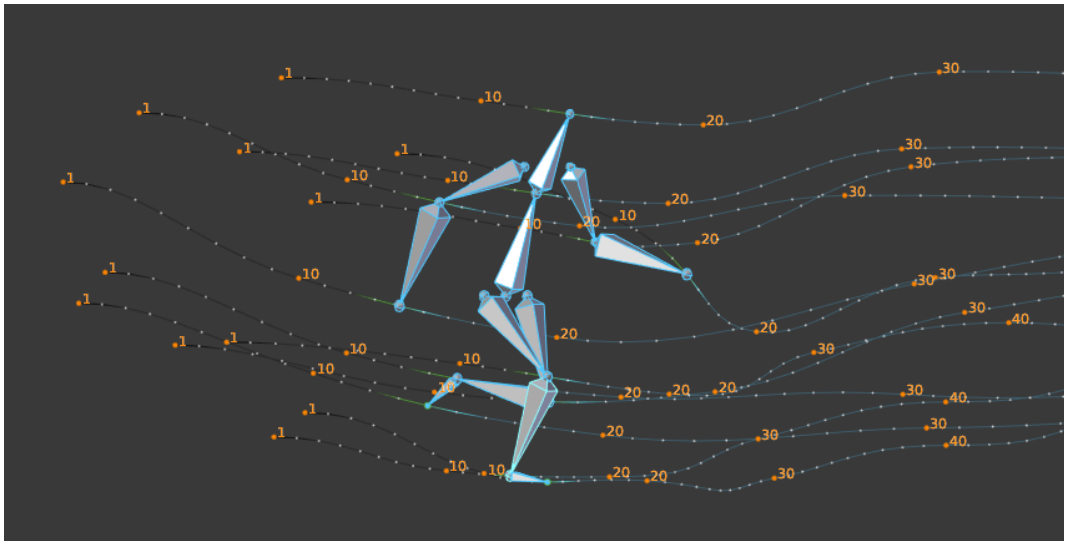
Thickness

Line thickness for motion path.

Custom Color

Use custom color for this motion path. The custom color can be modified for time before and after the current frame.

Example



An example of a motion path of an armature.

Physics

- Introduction
    - Quick Effects
- Rigid Body
    - Introduction
    - Properties
    - Rigid Body World
    - Rigid Body Constraints
    - Tips
- Cloth
    - Introduction
    - Settings
    - Examples
- Soft Body
    - Introduction
    - Settings
    - Forces
    - Collision
    - Examples
- Fluid
    - Introduction
    - Type
    - Materials
- Particle System
    - Introduction
    - Particle System Panel
    - Emitter
    - Hair
    - Texture Influence
    - Particle Edit Mode
- Dynamic Paint
    - Introduction
    - Brush
    - Canvas

- Forces
  - Gravity
  - Force Fields
- Collision
  - Options
  - Examples
  - Hints
- Baking Physics Simulations
  - Options
  - Baking
- Simulation Nodes
  - Baking
  - Examples

Introduction

Blender's physics system allows you to simulate a number of different real-world physical phenomena. You can use these systems to create a variety of static and dynamic effects such as:

- Hair, grass, and flocks
- Rain
- Smoke and dust
- Water
- Cloth
- Jello
- etc.

Quick Effects

Reference

Editor::	3D Viewport
Mode::	Object Mode
Menu::	*Object* ‣ *Quick Effects*

Sets up a basic simulation scene or effect including the selected objects. The tool will add essential objects like domains or particle systems both with predefined settings, so that there will be instant viewable result.

Quick Fur

Adds a fur setup to the selected objects. The fur setup is based on Geometry Nodes and built with Hair Node Groups that come with Blender as bundled assets.

Density
    Surface density of generated hair curves.

Length
: Length of the generated hair curves.

Hair Radius
: The width of the hair, used for rendering engines.

View Percentage
: Factor applied on the density for the viewport.

Apply Hair Curves
: Applies the modifier that uses the Generate Hair Curves node group.

Noise
: Deforms hair curves using a noise texture. See the Hair Curves Noise node group for more information.

Frizz
: Deforms hair curves using a random vector per point to frizz them. See the Frizz Hair Curves node group for more information.

Rigid Body

- Introduction
  - Creating a Rigid Body
  - Working with Rigid Bodies
- Properties
  - Settings
  - Collisions
  - Dynamics
- Rigid Body World
  - Settings
  - Rigid Body Cache
  - Rigid Body Field Weights
- Rigid Body Constraints
  - Introduction
  - Types
- Tips
  - Animation
  - Simulation Stability
  - Combining Rigid Bodies with Other Simulations
  - Scaling Rigid Bodies

Introduction

The rigid body simulation can be used to simulate the motion of solid objects. It affects the position and orientation of objects and does not deform them.

Unlike the other simulations in Blender, the rigid body simulation works closer with the animation system. This means that rigid bodies can be used like regular objects and be part of parent-child relationships, animation constraints and drivers.

Creating a Rigid Body

Reference

Mode::	All Modes
Panel::	*Properties ‣ Physics ‣ Rigid Body*
Menu::	*Object ‣ Rigid Body*

Only mesh objects can be part of a rigid body simulation. To create rigid bodies, either click on the *Rigid Body* button in the *Physics* tab of the Properties or use *Add Active/Add Passive* in the *Object ‣ Rigid Body* menu.

There are two types of rigid bodies: active and passive. *Active* bodies are dynamically simulated, while *passive* bodies remain static. Both types can be driven by the animation system when using the *Animated* option.

During the simulation, the rigid body system will override the position and orientation of dynamic rigid body objects. Note however, that the location and rotation of the objects are not changed, so the rigid body simulation acts similar to a constraint. To apply the rigid body transformations you can use the *Apply Object Transform* operator.

The scale of the rigid body object also influences the simulation, but is always controlled by the animation system.

Rigid body physics on the object can be *removed* with the *Rigid Body* button in the *Physics* tab in the Properties or in the *Object ▸ Rigid Body* menu.

Working with Rigid Bodies

Several object operators exist for working with rigid bodies, these operators can be found in the Rigid Body object menu. These operators include functions to add/remove rigid bodies, modify their properties, and add Rigid Body Constraints.

Properties

Reference

Panel::	*Physics ‣ Rigid Body*

Type

Role of the rigid body in the simulation.

Active::	The object is dynamic and is directly controlled by simulation results.
Passive::	The object remains static and is directly controlled by animation system, thus does not have Dynamics properties.

- Settings
- Collisions
  - Surface Response
  - Sensitivity
  - Collections
- Dynamics
  - Deactivation

Settings

Reference

Panel:: *Physics ▸ Rigid Body ▸ Settings*

Mass

Specifies how heavy the object is and “weights” irrespective of gravity.

Tip

There are predefined mass presets available with the Calculate Mass operator.

Dynamic

Enables/disables rigid body simulation for the object.

Animated

Allows the rigid body to additionally be controlled by the animation system.

Collisions

Reference

Panel::	*Physics* ▸ *Rigid Body* ▸ *Collisions*

Shape

Determines the collision shape of the object; these can be broken into two categories: primitive shapes and mesh based shapes.

Primitive shapes (*Box*, *Sphere*, *Capsule*, *Cylinder*, and *Cone*) are best in terms of memory and performance but do not necessarily reflect the actual shape of the object. They are calculated based on the object's bounding box. The center of gravity is always in the geometric center of the shape. Primitive shapes can be shown in the 3D Viewport by enabling Bounds.

Mesh based shapes (*Convex Hull* and *Mesh*) are calculated based on the geometry of the object so they are a better representation of the object. The center of gravity for these shapes is the object origin.

Box::	Box-like shapes (e.g. cubes), including planes (e.g. ground planes). The size per axis is calculated from the bounding box.
Sphere::	Sphere-like shapes. The radius is the largest axis of the bounding box.
Capsule::	This points up the Z axis.
Cylinder::	This points up the Z axis. The height is taken from the Z axis, while the radius is the larger of the X or Y axes.
Cone::	This points up the Z axis. The height is taken from the Z axis, while the radius is the larger of the X or Y axes.
Convex Hull::	

	A mesh-like surface encompassing (e.g. shrink-wrapped over) all vertices (best results with fewer vertices). A convex approximation of the object, which has good performance and stability.
Mesh::	Mesh consisting of triangles only, allowing for more detailed interactions than convex hulls. Allows simulating concave objects, but is rather slow and unstable.
Compound Parent::	Takes the collision shapes from the object's children and combines them. This makes it possible to create concave shapes from primitive shapes. This usually results in a faster simulation than the *Mesh* collision shape while also being generally more stable.

Source

Source of the mesh used to create the collision shape.

Base::	The base mesh of the object.
Deform::	Includes any deformations added to the mesh (shape keys, deform modifiers).
Final::	Includes all deformations and modifiers.

Deforming

Mesh shapes can deform during simulation.

Surface Response

Friction

Resistance of object to movement. Specifies how much velocity is lost when objects collide with each other.

Bounciness

Tendency of object to bounce after colliding with another (0 to 1) (rigid to perfectly elastic). Specifies how much objects can bounce after collisions.

Sensitivity

The collision margin is used to improve the performance and stability of rigid bodies. Depending on the shape, it behaves differently: some shapes embed it, while others have a visible gap around them.

The margin is *embedded* for these shapes:

- Sphere
- Box
- Capsule
- Cylinder
- Convex Hull: Only allows for uniform scale when embedded.

The margin is *not embedded* for these shapes:

- Cone
- Active Triangle Mesh
- Passive Triangle Mesh: Can be set to 0 most of the time.

Margin

> Threshold of distance near the surface where collisions are still considered (best results when nonzero).

Collections

Allows rigid body collisions allocate on different groups (maximum 20).

Dynamics

Reference

Panel:: *Physics ▸ Rigid Body ▸ Dynamics*

Used to control the physics of the rigid body simulation. This panel is available only for *Active* type of rigid bodies.

Damping Translation
Amount of linear velocity that is lost over time.

Rotation
Amount of angular velocity that is lost over time.

Deactivation

Enable deactivation of resting rigid bodies. Allows the object to be deactivated during the simulation (improves the performance and stability, but can cause glitches).

Start Deactivated
The rigid body starts deactivated. It will be activated when in proximity of moving active rigid body objects. The proximity check uses the object's bounding box to determine if a moving object is close enough to activate it.

Linear Velocity
Specifies the linear deactivation velocity below which the rigid body is deactivated and the simulation stops simulating the object.

Angular Velocity
Specifies the angular deactivation velocity below which the rigid body is deactivated and the simulation stops simulating the object.

Rigid Body World

Reference

Panel:: *Scene ▸ Rigid Body World*

The *Rigid Body World* is a group of rigid body objects, which holds settings that apply to all rigid bodies in this simulation.

When you add rigid body physics to an object, primary there is created a group of objects with default “RigidBodyWorld” name. Rigid body objects automatically are added to this group when you add rigid body physics for them. You can create several Rigid Body World Collections and allocate the rigid body objects with the Collections panel.

Rigid body objects and constraints are only taken into account by the simulation if they are in the collection specified in the *Collection* field of the *Rigid Body World* panel in the *Scene* tab.

Settings

Rigid Body World
: Enable/disable evaluation of the rigid body simulation based on the rigid body objects participating in the specified group of Rigid Body World.

Remove Rigid Body World
: Remove rigid body simulation from the current scene.

Collection
: Containing rigid body objects participating in this simulation.

Constraints
: Containing rigid body object constraints participating in the simulation.

Simulation quality and timing settings:

Speed
  Can be used to speed up/slow down the simulation.

Split Impulse
  Enable/disable reducing extra velocity that can build up when objects collide (lowers the simulation stability a little so use only when necessary). Limits the force with which objects are separated on collision, generally produces nicer results, but makes the simulation less stable (especially when stacking many objects).

Substeps Per Frame
  Number of simulation steps taken per frame (higher values are more accurate but slower). This only influences the accuracy and not the speed of the simulation.

Solver Iterations
  Amount of constraint solver iterations made per simulation step (higher values are more accurate but slower). Increasing this makes constraints and object stacking more stable.

Rigid Body Cache

Reference

Panel::	*Scene* ‣ *Rigid Body World* ‣ *Cache*

The *Cache* subpanel specifies the frame range in which the simulation is active. Can be used to bake the simulation.

Start/End
  First and last frame of the simulation.

Bake
  Calculates the simulation and protects the cache. You need to be in *Object Mode* to bake.

Delete Bake
  Active after the baking of simulation. Clears the baked cache.

Calculate to Frame
  Bake physics to current frame.

Current Cache to Bake
  Bake from Cache.

Bake All Dynamics
  Bake all physics.

Delete All Bakes
  Deletes all baked caches of all objects in the current scene.

Update All to Frame
  Update cache to current frame.

If you have not saved the blend-file, the cache is created in memory, so save your file first or the cache may be lost.

Rigid Body Field Weights

Reference

Panel::	*Scene ‣ Rigid Body World ‣ Field Weights*

As other physics dynamics systems, rigid body simulation are also influenced by external force effectors.

Rigid Body Constraints

- Introduction
  - Connect
  - Physics Menu
  - Common Options
- Types
  - Fixed Constraint
  - Point Constraint
  - Hinge Constraint
  - Slider Constraint
  - Piston Constraint
  - Generic Constraint
  - Generic Spring Constraint
  - Motor Constraint

Introduction

Constraints (also known as joints) for rigid bodies connect two rigid bodies. The physics constraints are meant to be attached to an Empty object. The constraint then has fields which can be pointed at the two physics-enabled object which will be bound by the constraint. The empty object provides a location and axis for the constraint distinct from the two constrained objects. The location of the entity hosting the physics constraint marks a location and set of axes on each of the two constrained objects. These two anchor points are calculated at the beginning of the animation and their position and orientation remain fixed in the *local* coordinate system of the object for the duration of the animation. The objects can move far from the constraint object, but the constraint anchor moves with the object. If this feature seems limiting, consider using multiple objects with a non-physics *Child of* constraint and animate the relative location of the child.

Connect

The quickest way to constrain two objects is to select both and click the *Connect* button in *Object* ▸ *Rigid Body*. This creates a new empty object (named “Constraint”) with a physics constraint already attached and pointing at the two selected objects.

Physics Menu

Also you can create *Rigid Body Constraint* on one of the two constrained objects with *Rigid Body Constraint* button of the *Physics* tab in the Properties. This constraint is dependent on the object location and rotation on which it was created. This way, there are no empty object created for the constraint. The role of the empty object is put on this object. The constrained object can be then be set as a *Passive* type for better driving of the constraint.

Additional parameters appear in the *Rigid Body Constraint* panel of the *Physics* tab in the Properties for the selected empty object or the one of the two constrained objects with the created constraint.

Common Options

Reference

Panel:: *Physics* ‣ *Rigid Body Constraint*

Settings

Enabled
: Specifies whether the constraint is active during the simulation.

Disable Collisions
: Allows constrained objects to pass through one another.

Breakable
: Allows constraint to break during simulation. Disabled for the *Motor* constraint. This can be used to simulate destruction.

Threshold
: Impulse strength that needs to be reached before the constraint breaks.

Limits

By using limits you can constrain objects even more by specifying a translation/rotation range on/around respectively one axis (see below for each one individually). To lock one axis, set both limits to 0.

Objects

First
: First object to be constrained.

Second
: Second object to be constrained.

Override Iterations

Allows making constraints stronger (more iterations) or weaker (less iterations) than specified in the rigid body world.

Iterations
: Number of constraint solver iterations made per simulation step for this constraint.

Types

- Fixed Constraint
- Point Constraint
- Hinge Constraint
    - Options
- Slider Constraint
    - Options
- Piston Constraint
    - Options
- Generic Constraint
    - Options
- Generic Spring Constraint
    - Options
- Motor Constraint
    - Options

Fixed Constraint

Reference

Panel::	*Physics* ‣ *Rigid Body Constraint*
Type::	Fixed

This constraint cause the two objects to move as one. Since the physics system does have a tiny bit of slop in it, the objects do not move as rigidly as they would if they were part of the same mesh.

Point Constraint

Reference

Panel::	*Physics* ‣ *Rigid Body Constraint*
Type::	Point

The objects are linked by a point bearing allowing any kind of rotation around the location of the constraint object, but no relative translation is permitted. The physics engine will do its best to make sure that the two points designated by the constraint object on the two constrained objects are coincident.

Hinge Constraint

Reference

Panel::	*Physics* ▸ *Rigid Body Constraint*
Type::	Hinge

The hinge permits one degree of freedom between two objects. Translation is completely constrained. Rotation is permitted about the Z axis of the object hosting the Physics constraint (usually an Empty, distinct from the two objects that are being linked). Adjusting the position and rotation of the object hosting the constraint allows you to control the anchor and axis of the hinge.

The Hinge is the only single-axis rotational constraint that uses the Z axis instead of the X axis. If something is wrong with your hinge, check your other constraints to see if this might be the problem.

Options

Limits
- Z Angle
  - Enables/disables limit rotation around Z axis.
  - Lower
    - Lower limit of Z axis rotation.
  - Upper
    - Upper limit of Z axis rotation.

Slider Constraint

Reference

Panel::	*Physics ‣ Rigid Body Constraint*
Type::	Slider

The Slider constraint allows relative translation along the X axis of the constraint object, but permits no relative rotation, or relative translation along other axes.

Options

Limits

- X Axis
  - Enables/disables limit translation around X axis.
  - Lower
    - Lower limit of X axis translation.
  - Upper
    - Upper limit of X axis translation.

Piston Constraint

Reference

Panel::	*Physics ‣ Rigid Body Constraint*
Type::	Piston

A piston permits translation along the X axis of the constraint object. It also allows rotation around the X axis of the constraint object. It is like a combination of the freedoms of a slider with the freedoms of a hinge (neither of which is very free alone).

Options

Limits
- X Axis
  - Enables/disables limit translation around X axis.
  - Lower
    - Lower limit of X axis translation.
  - Upper
    - Upper limit of X axis translation.
- X Angle
  - Enables/disables limit rotation around X axis.
  - Lower
    - Lower limit of X axis rotation.
  - Upper
    - Upper limit of X axis rotation.

Generic Constraint

Reference

Panel::	*Physics ‣ Rigid Body Constraint*
Type::	Generic

The generic constraint has a lot of available parameters.

The X, Y, and Z axis constraints can be used to limit the amount of translation between the objects. Clamping the min/max to zero has the same effect as the Point constraint.

Clamping the relative rotation to zero keeps the objects in alignment. Combining an absolute rotation and translation clamp would behave much like the Fixed constraint.

Using a nonzero spread on any parameter allows it to oscillate in that range throughout the course of the simulation.

Options

Limits

  Angular

    X Angle, Y Angle, Z Angle

      Enables/disables limit rotation around X, Y or Z axis respectively.

      Lower

        Lower limit of rotation for X, Y or Z axis respectively.

      Upper

        Upper limit of rotation for X, Y or Z axis respectively.

Linear

X Axis, Y Axis, Z Axis

Enables/disables limit translation on X, Y or Z axis respectively.

Lower

Lower limit of translation for X, Y or Z axis respectively.

Upper

Upper limit of translation for X, Y or Z axis respectively.

Generic Spring Constraint

Reference

Panel::	*Physics ‣ Rigid Body Constraint*
Type::	Generic Spring

The generic spring constraint adds some spring parameters for the X/Y/Z axes to all the options available on the Generic constraint. Using the spring alone allows the objects to bounce around as if attached with a spring anchored at the constraint object. This is usually a little too much freedom, so most applications will benefit from enabling translation or rotation constraints.

If the damping on the springs is set to 1, then the spring forces are prevented from realigning the anchor points, leading to strange behavior. If your springs are acting weird, check the damping.

Options

Limits

- X/Y/Z Axis
  - Enables/disables limit translation on X, Y or Z axis respectively.
  - Lower
    - Lower limit of translation for X, Y or Z axis respectively.
  - Upper
    - Upper limit of translation for X, Y or Z axis respectively.
- X/Y/Z Angle
  - Enables/disables limit rotation around the X, Y or Z axis respectively.

Lower

Lower limit of rotation for X, Y or Z axis respectively.

Upper

Upper limit of rotation for X, Y or Z axis respectively.

Springs

X/Y/Z Axis

Enables/disables springs translation on X, Y or Z axis respectively.

Stiffness

Spring Stiffness of the translation on X, Y or Z axis respectively. Specifies how “bendy” the spring is.

Damping

Spring Damping of the translation on X, Y or Z axis respectively. Amount of damping the spring has.

X/Y/Z Angle

Enables/disables springs rotation around the X, Y or Z axis respectively.

Stiffness

Spring Stiffness of the rotation around the X, Y or Z axis respectively. Specifies how “bendy” the spring is.

Damping

Spring Damping of the rotation around the X, Y or Z axis respectively. Amount of damping the spring has.

Motor Constraint

Reference

Panel::	*Physics ‣ Rigid Body Constraint*
Type::	Motor

The motor constraint causes translation and/or rotation between two entities. It can drive two objects apart or together. It can drive simple rotation, or rotation and translation (although it will not be constrained like a screw since the translation can be blocked by other physics without preventing rotation).

The rotation axis is the X axis of the object hosting the constraint. This is in contrast with the Hinge which uses the Z axis. Since the Motor is vulnerable to confusing perturbations without a matching Hinge constraint, special care must be taken to align the axes. Without proper alignment, the motor will appear to have no effect (because the hinge is preventing the motion of the motor).

Options

Linear Motor/Angular Motor

Enable

Enable linear or angular motor respectively.

Target Velocity

Target linear or angular motor velocity respectively.

Max Impulse

Maximum linear or angular motor impulse respectively.

Tips

As with all physics-enabled objects, pay close attention to the *Animated* checkbox in the *Rigid Body* panel of the *Physics* tab in the Properties. A common mistake is to use keyframe animation on a *Passive* physics object without checking the *Animated* box. The object will move, but the physics engine will behave as if the *Passive* is still in its starting place, leading to disappointment.

Animation

The most common trick is to Keyframe animate the location or rotation of an *Active* physics object as well as the *Animated* checkbox. When the curve on the *Animated* property switches to disabled, the physics engine takes over using the object's last known location, rotation and velocities.

Animating the strengths of various other parameters (a Motor's Target Velocity, a Hinge's limits, etc.) can be used to accomplish a wide variety of interesting results.

Enabling a constraint during the physics simulation often has dramatic results as the physics engine tries to bring into alignment two objects which are often dramatically out of alignment. It is very common for the affected objects to build up enough kinetic energy to bounce themselves out of camera.

Rigid body dynamics can be baked to normal keyframes with *Bake To Keyframes* in the *Object* ▸ *Rigid Body* menu.

Simulation Stability

The simplest way of improving simulation stability is to increase the steps per second. However, care has to be taken since making too many steps can cause problems and make the simulation even less stable (if you need more than 1000 steps, you should look at other ways to improve stability).

Increasing the number of solver iterations helps making constraints stronger and also improves object stacking stability.

It is best to avoid small objects, as they are currently unstable. Ideally, objects should be at least 20 cm in diameter. If it is still necessary, setting the collision margin to 0, while generally not recommended, can help making small object behave more naturally.

When objects are small and/or move very fast, they can pass through each other. Besides what is mentioned above it's also good to avoid using mesh shapes in this case. Mesh shapes consist of individual triangles and therefore do not really have any thickness, so objects can pass through more easily. You can give them some thickness by increasing the collision margin.

Combining Rigid Bodies with Other Simulations

Since the rigid body simulation is part of the animation system, it can influence other simulations just like the animation system can.

In order for this to work, the rigid body object needs to have a Collision Modifier. Simply click on *Collision* in the *Physics* tab.

Scaling Rigid Bodies

Rigid body objects can be scaled, also during the simulation. This work well in most cases, but can sometimes cause problems.

If dynamic scaling is not needed, rigid body objects should have the scale applied by using the *Apply Scale* tool `Ctrl-A`.

Cloth

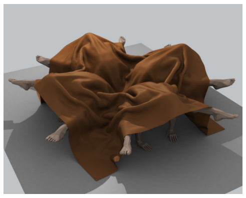
- Introduction
  - Workflow
  - Springs
- Settings
  - Physical Properties
  - Cache
  - Shape
  - Collisions
  - Property Weights
  - Field Weights
- Examples
  - Using Simulation to Shape/Sculpt a Mesh
  - Smoothing of Cloth
  - Cloth on Armature
  - Cloth with Animated Vertex Groups
  - Cloth with Dynamic Paint
  - Using Cloth for Soft Bodies
  - Cloth with Wind

Introduction

Cloth simulation is one of the hardest aspects of computer graphics, it is a deceptively simple real-world item that is taken for granted, but it actually has very complex internal and environmental interactions. Cloth is commonly modeled as 2D mesh to simulate real world objects such as fabrics, flags, banners. And yet cloth can also be used to model 3D objects such as teddy bears, pillows, balloons, or balls.

Cloth interacts with and is affected by other moving objects, the wind and other forces, as well as a general aerodynamic model, all of which is under your control.

Cloth example.



Cloth on carved wooden men (made by motorsep).

Cloth example.

Once Cloth physics have been added to a mesh, a Cloth Modifier will be added to the object's modifier stack. As a modifier then, it can interact with other modifiers, such as *Armature* and *Smooth*. In these cases, the ultimate shape of the mesh is computed in accordance with the order of the modifier stack. For example, you should smooth the cloth *after* the modifier computes the shape of the cloth.

You can *Apply* the Cloth Modifier to freeze, or lock in, the shape of the mesh at that frame, which removes the modifier. For example, you can drape a flat cloth over a table, let the simulation run, and then apply the modifier. In this sense, you are using the simulator to save yourself a lot of modeling time.

Results of the simulation are saved in a cache, so that the shape of the mesh, once calculated for a frame in an animation, does not have to be recomputed again. If changes to the simulation are made, you have full control over clearing the cache and re-running the simulation. Running the simulation for the first time is fully automatic and no baking or separate step interrupts the workflow.

Computation of the shape of the cloth at every frame is automatic and done in the background; thus you can continue working while the simulation is computed. However, it is CPU-intensive and depending on the power of your PC and the complexity of the simulation, the amount of CPU needed to compute the mesh varies, as does the lag you might notice.

Note

Do Not Jump Ahead

If you set up a cloth simulation but Blender has not computed the shapes for the duration of the simulation, and if you jump ahead a lot of frames forward in your animation, the cloth simulator may not be able to compute or show you an accurate mesh shape for that frame, if it has not previously computed the shape for the previous frame(s).

Workflow

A general process for working with cloth is to:

1. Model the cloth object as a general starting shape.
2. Designate the object as a “cloth” in the *Physics* tab of the Properties.
3. Model other deflection objects that will interact with the cloth. Ensure the Deflection modifier is last on the modifier stack, after any other mesh deforming modifiers.

4. Light the cloth and assign materials and textures, UV unwrapping if desired.
5. If desired, give the object particles, such as steam coming off the surface.
6. Run the simulation and adjust settings to obtain satisfactory results. The Timeline editors playback controls are great for this step.
7. Optionally age the mesh to some point in the simulation to obtain a new default starting shape.
8. Make minor edits to the mesh on a frame-by-frame basis to correct minor tears.

Tip

To avoid unstable simulation, make sure that the cloth object does not penetrate any of the deflection objects.

Springs

Internally, cloth physics is simulated with virtual springs that connect the vertices of a mesh. There are four types of springs that control how the cloth bends. These four types are defined below and illustrated in the following image:

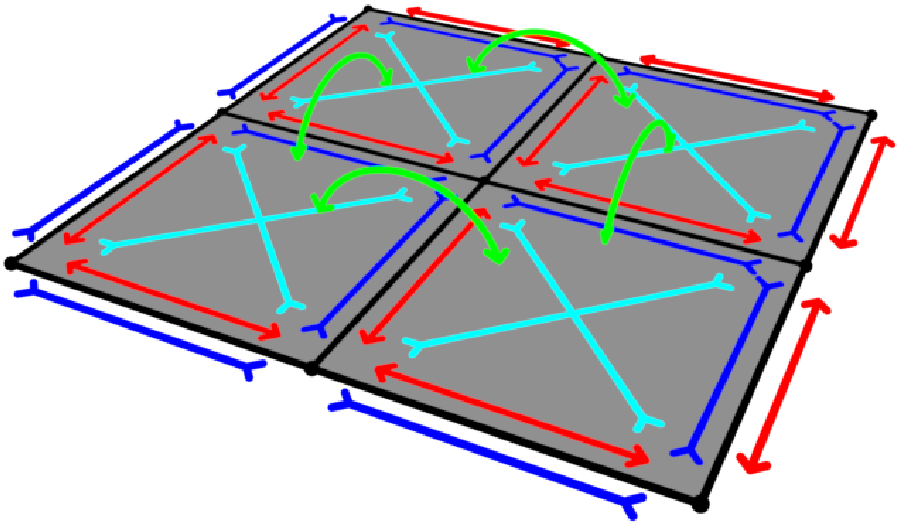


Illustration of cloth springs; tension springs (blue), compression springs (red), shear springs (cyan), and angular bending springs (green).

Tension Springs
: Control the stiffness of the cloth.

Compression Springs
: Control the amount of force required to collapse or compress the cloth.

Shear Springs
: Like compression springs but it controls the angular deformation.

Angular Bending Springs
: Control how resilient the cloth is to folding or crumpling.

All four of these spring types can be controlled independently in the Physical Properties panel. While these settings control surface springs, optionally, internal springs can be used for 3D meshes and behave similarly to Soft Bodies.

Settings

Reference

Panel:: *Physics ‣ Cloth*

Presets
: Contains a number of preset cloth examples.

Quality Steps
: Set the number of simulation steps per frame. Higher values result in better quality, but will be slower.

Speed Multiplier
: Adjust how fast time progresses in the cloth simulation.

- Physical Properties
  - Stiffness
  - Damping
  - Internal Springs
  - Pressure
- Cache
  - Editing the Cached Simulation
- Shape
- Collisions
  - Object Collisions
  - Self-Collisions
- Property Weights
- Field Weights

Physical Properties

Reference

Panel:: *Physics ▸ Cloth ▸ Physical Properties*

Vertex Mass
The mass of the cloth material.

Air Viscosity
Air has some thickness which slows falling things down.

Bending Model

Linear::	Cloth model with linear bending springs (old).
Angular::	Cloth model with angular bending springs.

Stiffness

Tension
How much the material resists stretching.

Compression
How much the material resists compression.

Structural
Overall stiffness of the cloth (only in linear bending model).

Shear
How much the material resists shearing.

Bending
Wrinkle coefficient. Higher creates more large folds.

Damping

Tension
: Amount of damping in stretching behavior.

Compression
: Amount of damping in compression behavior.

Structural
: Amount of damping in stretching behavior (only in linear bending model).

Shear
: Amount of damping in shearing behavior.

Bending
: Amount of damping in bending behavior.

Internal Springs

As stated in the introduction, cloth physics are simulated through Springs connecting vertices on the surface of a mesh. But these springs only interact on the surface and only apply to 2D surfaces. 3D or *Internal Springs* can be used to make a mesh behave similarly to a Soft Body. Internal springs can be enabled by toggling the checkbox in the *Internal Springs* panel header.

Max Spring Creation Length
: The maximum length an internal spring can have during creation. If the distance between internal points is greater than this, no internal spring will be created between these points. A length of zero means that there is no length limit.

Max Creation Diversion
: The maximum angle that is allowed to use to connect the internal points can diverge from the vertex normal.

Check Surface Normals
: Requires the points the internal springs connect to have opposite normal directions.

Tension
  How much the material resists stretching.

Compression
  How much the material resists compression.

Vertex Group
  The *Tension* and *Compression* of internal springs can be controlled via a Vertex Group to specify which the portions of the mesh have internal springs or the spring strength.

Max Tension
  Maximum tension stiffness value.

Max Compression
  Maximum Compression stiffness value.

Pressure

Cloth pressure allows the simulation of soft-shelled objects such as balloons or balls that are filled with a type of fluid. This fluid is modeled as a gas; to emulate an incompressible liquid set *Pressure Scale* as high as possible without breaking the simulation. Cloth pressure can be enabled by toggling the checkbox in the *Pressure* panel header.

Note

Non-manifold meshes will work with cloth pressure however, pressure will escape out of the mesh holes and cause drifting or propulsion forces. One way to get around this is by using the *Vertex Group* to exclude the non-manifold portions of the mesh.

Pressure
  The uniform pressure that is constantly applied to the mesh. This value is specified in units of *Pressure Scale*, and can be negative to simulate

implosions or any other case where an object has outside pressure pushing inwards.

Custom Volume

Use the *Target Volume* parameter as the initial volume for the cloth, instead of computing it from the mesh itself.

Target Volume

The mesh volume where the inner/outer pressure will be the same. If set to zero, changes in the volume of the object will not affect pressure.

Pressure Scale

Ambient pressure (in kPa) that exists both inside and outside the object, balancing out when the volume matches the target. Increase the value to make the object resist changes in volume more strongly.

Fluid Density

Specifies the density of the fluid contained inside the object (in kg/liter = 1000 kg/m^3, use 1 for water), used to generate a hydrostatic pressure gradient that simulates the weight of the fluid. If the value is negative, it instead models buoyancy from a surrounding fluid.

The fluid is not actually simulated, so while the setting helps to achieve a more plausible object shapes at rest, it cannot create realistic fluid dynamics effects. It can also be used to give more weight to a soft body like object with heavy and sufficiently flexible filling, even if it is not a fluid by itself.

The volume of the object is not preserved. If that is desired it should be used together with *Pressure Scale*. *Fluid density* times *object size* times 50 is a good start value for *Scale* to make sure that no more than 10% volume change if the object does not experience higher acceleration than standard gravity.

Vertex Group

Cloth pressure can be controlled via a Vertex Group to specify which the portions of the mesh to apply pressure. Zero weight means no pressure while a weight of one means full pressure.

Note, faces with a vertex that has zero weight will be excluded from the *Target Volume* calculation.

Cache

Reference

Panel:: *Physics ‣ Cloth Cache*

After you have set up the deflection mesh for the frame range you intend to run the simulation (including animating that mesh *via* armatures), you can now tell the cloth simulation to compute (and avoid) collisions. Select the cloth object and in the *Object* tab, *Physics* tab, set the *Start* and *End* settings for the simulation frames you wish to compute, and click the *Bake* button.

Cache settings for cloth are the same as with other dynamic systems. See Particle Cache for details.

Note

If you move or edit the cloth object **after** you have already run the simulations, you must clear the cache; otherwise, Blender will use the position of the current/cached mesh's vertices when trying to represent where they are.

Note

Subdivision Surface Modifier

A bake/cache is done for every subdivision level so please use the **equal** subdivision level for render and preview.

Note

You cannot change *Start* or *End* without clearing the bake simulation. When the simulation has finished, you will notice you have the option to free the bake, edit the bake and re-bake.

Editing the Cached Simulation

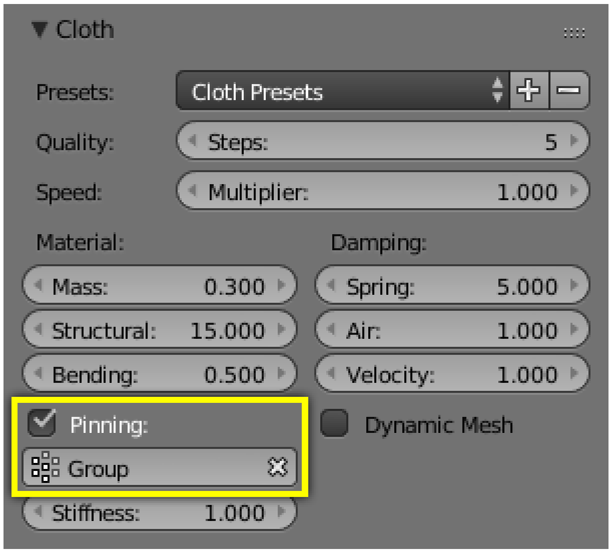
Important

Editing the cached simulation is not currently working, see: blender/blender#77114 for details.

Shape

Reference

Panel:: *Physics ▸ Cloth ▸ Shape*



Presets: Cloth Presets

Quality: Steps: 5

Speed: Multiplier: 1.000

Material:

Mass: 0.300

Structural: 15.000

Bending: 0.500

☑ Pinning:

Group

Stiffness: 1.000

Damping:

Spring: 5.000

Air: 1.000

Velocity: 1.000

☐ Dynamic Mesh

Cloth Shape.

Pin Group
: Vertex group to use for pinning.

  The shape of the cloth can be controlled by pinning cloth to a Vertex Group. There are several ways of doing this including Weight Painting areas you want to pin. The weight of each vertex in the group controls how strongly it is pinned.

Stiffness
: Target position stiffness.

Sewing
: Another method of restraining cloth similar to pinning is sewing springs. Sewing springs are virtual springs that pull vertices in one part

of a cloth mesh toward vertices in another part of the cloth mesh. This is different from pinning which binds vertices of the cloth mesh in place or to another object. A clasp on a cloak could be created with a sewing spring. The spring could pull two corners of a cloak about a character's neck. This could result in a more realistic simulation than pinning the cloak to the character's neck since the cloak would be free to slide about the character's neck and shoulders.

Sewing springs are created by adding extra edges to a cloth mesh that are not included in any faces. They should connect vertices in the mesh that should be pulled together. For example the corners of a cloak.

Max Sewing Force

Maximum force that can be applied by sewing springs. Zero means unbounded, but it is not recommended to leave the field at zero in most cases, as it can cause instability due to extreme forces in the initial frames where the ends of the sewing springs are far apart.

Shrinking Factor

Factor by which to shrink the cloth, specifying a negative value controls the amount for the cloth to grow.

Dynamic Mesh

Allows animating the rest shape of cloth using shape keys or modifiers (e.g. an Armature modifier or any deformation modifier) placed above the Cloth modifier. When it is enabled, the rest shape is recalculated every frame, allowing unpinned cloth to squash and stretch following the character with the help of shape keys or modifiers, but otherwise move freely under control of the physics simulation.

Normally cloth uses the state of the object in the first frame to compute the natural rest shape of the cloth, and keeps that constant throughout the simulation. This is reasonable for fully realistic scenes, but does not quite work for clothing on cartoon style characters that use a lot of squash and stretch.

Rest Shape Key

Allows starting the cloth simulation using a specific Shape Key as the rest state, instead of the shape that results from evaluating shape keys and preceding modifiers in the regular way. This option is mutually exclusive with *Dynamic Mesh*.

This can be used to start the simulation with the cloth in a pre-draped state without applying that shape as a plastic deformation that relaxes all springs as a side effect.

This property is only visible if the mesh has shape keys.

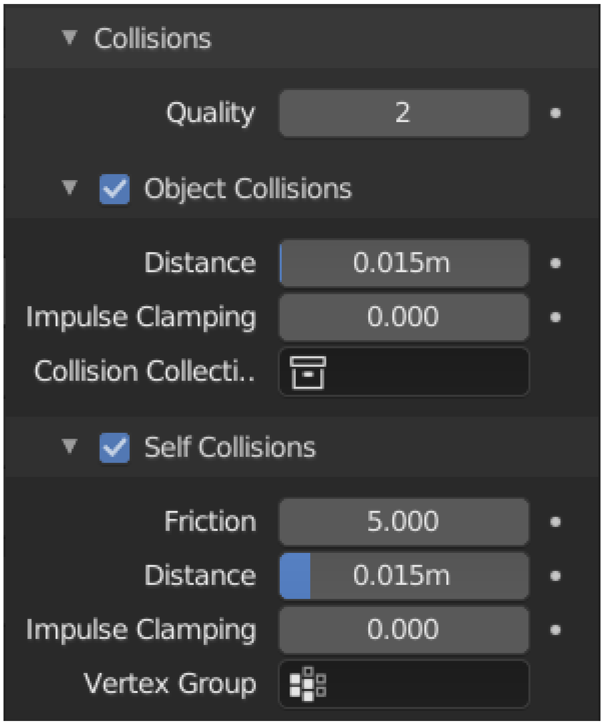
Collisions

Reference

Panel:: *Physics* ▸ *Cloth* ▸ *Collision*

In most cases, a piece of cloth does not just hang there in 3D space, it collides with other objects in the environment. To ensure proper simulation, there are several items that have to be set up and working together:

- The *Cloth* object must be told to participate in collisions.
- Optionally (but recommended) tell the cloth to collide with itself.
- Other objects must be visible to the *Cloth* object *via* shared layers.
- The other objects must be mesh objects.
- The other objects may move or be themselves deformed by other objects (like an armature or shape key).
- The other mesh objects must be told to deflect the cloth object.
- The blend-file must be saved in a directory so that simulation results can be saved.
- You then *Bake* the simulation. The simulator computes the shape of the cloth for a frame range.
- You can then edit the simulation results, or make adjustments to the cloth mesh, at specific frames.
- You can make adjustments to the environment or deforming objects, and then re-run the cloth simulation from the current frame forward.



Collisions
Quality 2
☑ Object Collisions
Distance 0.015m
Impulse Clamping 0.000
Collision Collecti..
☑ Self Collisions
Friction 5.000
Distance 0.015m
Impulse Clamping 0.000
Vertex Group

Cloth Collisions panel.

Quality

A general setting for how fine and good a simulation you wish. Higher numbers take more time but ensure less tears and penetrations through the cloth.

Object Collisions

If the cloth object needs to be deflected by some other object. To deflect a cloth, the object must be enabled as an object that collides with the cloth object. To enable objects to collide with cloth objects enable collision physics for the collider object (not on the cloth object).

Note

If your colliding object is not a mesh object, such as a NURBS surface, or a text object, you must convert it to a mesh object using Convert.

Distance

The distance another object must get to the cloth for the simulation to repel the cloth out of the way. Smaller values might give errors but gives some speed-up while larger will give unrealistic results if too large and can be slow. It is best to find a good in between value.

Impulse Clamping

Prevents explosions in tight and complicated collision situations by restricting the amount of movement after a collision.

Vertex Group

Faces that have all vertices assigned to this Vertex Group are excluded from collision with objects.

Collision Collection

Only objects that are a part of this Collection can collide with the cloth. Note that these objects must also have Collision physics enabled.

Self-Collisions

Real cloth cannot penetrate itself, so you normally want the cloth to self-collide. Enable this to tell the cloth object that it should not penetrate itself. This adds to the simulation's compute time, but provides more realistic results.

Tip

A flag, viewed from a distance does not need this enabled, but a close-up of a cape or blouse on a character should have this enabled.

Friction

A coefficient for how slippery the cloth is when it collides with itself. For example, silk has a lower coefficient of friction than cotton.

Distance

As cloth at this distance begins to repel away from itself. Smaller values might give errors but gives some speed-up while larger will give unrealistic results if too large and can be slow. It is best to find a good in between value.

Impulse Clamping

Prevents explosions in tight and complicated collision situations by restricting the amount of movement after a collision.

Vertex Group

Faces that have all vertices assigned to this Vertex Group are excluded from self-collision.

See also

Example blend-file: Cloth self-collisions.

Troubleshooting

If you encounter some problems with collision detection, there are a few ways to fix them:

- The fastest solution is to increase the *Distance* for Object/Self Collisions. This will be the fastest way to fix the clipping; however, it will be less accurate and will not look as good. Using this method tends to make it look like the cloth is resting on air, and gives it a very rounded look.
- A second method is to increase the *Quality* (in the *Cloth* panel). This results in smaller steps for the simulator and therefore to a higher probability that fast-moving collisions get caught. You can also increase the Collisions *Quality* to perform more iterations to get collisions solved.
- If none of the methods help, you can easily edit the cached/baked result in *Edit Mode* afterwards.
- If the Cloth is torn by the deforming mesh; increase the stiffness settings.

Property Weights

Reference

Panel:: *Physics ‣ Cloth ‣ Property Weights*

This panel is used to constrain certain cloth properties to a certain vertex group. The properties that they control can be found in a combination of the *Physical Properties* and *Shape* panels.

Structural Group	Defines a vertex group to control over structural stiffness.
Max Tension	Maximum tension stiffness value.
Max Compression	Maximum Compression stiffness value.
Shear Group	Vertex group for fine control over shear stiffness.
Max Shearing	Maximum shear scaling value.
Bending Group	Vertex group for fine control over bending stiffness.
Max Bending	Maximum bending stiffness value.
Shrinking Group	Vertex group for shrinking cloth.
Max Shrinking	

Max amount to shrink cloth by, specifying a negative value controls the max amount for the cloth to grow.

Field Weights

Reference

Panel:: *Physics* ▸ *Cloth* ▸ *Field Weights*

As other physics dynamics systems, Cloth simulation is also influenced by external force effectors.

Examples

To start with cloth, the first thing you need, of course, is some fabric. So, let us delete the default cube and add a plane. In order to get some good floppy and flexible fabric, you will need to subdivide it several times, about eight is a good number. So `Tab` into *Edit Mode* and subdivide the mesh a couple of times.

Now, we will make this cloth by going to the Physics tab. Scroll down until you see the *Cloth* panel, and press the *Cloth* button. Now, a lot of settings will appear, most of which we will ignore for now.

That is all you need to do to set your cloth up for animating, but if you playback the animation, the drop of your newly created fabric will be quite unspectacular. That is what we will cover in the next two sections about pinning and colliding.

Using Simulation to Shape/Sculpt a Mesh

You can *Apply* the Cloth Modifier at any point to freeze the mesh in position at that frame. You can then re-enable the cloth, setting the start and end frames from which to run the simulation forward.

Another example of aging is a flag. Define the flag as a simple grid shape and pin the edge against the flagpole. Simulate for 50 frames or so, and the flag will drop to its “rest” position. Apply the *Cloth* Modifier. If you want the flag to flap or otherwise move in the scene, re-enable it for the frame range when it is in camera view.

Smoothing of Cloth

Now, if you followed this from the previous section, your cloth is probably looking a little blocky. In order to make it look nice and smooth like the picture you need to apply a Smooth and/or Subdivision Surface Modifier in the *Modifiers* tab. Then, in the Toolbar, find the *Edit* panel and Press *Smooth*.

Cloth on Armature

Clothing can be simulated and pinned to an armature. For example, a character could have a baggy tunic pinned to the character's waist with a belt.

The typical workflow for pinning:

1. Set the armature to its bind pose.
2. Model clothing that encloses but does not penetrate the character's mesh.
3. Parent the clothing objects to the armature. The armature will now have several child meshes bound to it.
4. Create a new vertex group on each cloth object for its pinned vertices.
5. Add vertices to be pinned to this vertex group and give these vertices nonzero weights (you probably want weight = 1). For example the belt area of the tunic would be in the vertex group and have weight one.
6. Designate the clothing objects as "cloth" in the Physics tab of the Properties. Make sure the Cloth Modifier is below the Armature Modifier in the modifier stack.
7. In the cloth Shape panel select the vertex group.
8. Add collision physics to the character's mesh.
9. The clothing is now ready; non-pinned vertices will be under control of the Cloth modifier. Pinned vertices will be under control of the Armature modifier.

Note

When animating or posing the character you must begin from the bind pose. Move the character to its initial pose over several frames so the physics engine can simulate the clothing moving. Very fast movements and teleport jumps can break the physics simulation.

Regression blend-file.

Cloth with Animated Vertex Groups

Cloth with animated pinned vertices: Regression blend-file. Unsupported: Starting with a goal of 0 and increasing it, but still having the vertex not pinned will **not** work (e.g. from goal = 0 to goal = 0.5).

Cloth with Dynamic Paint

Cloth with Dynamic Paint using animated vertex groups: Regression blend-file. Unsupported: Starting with a goal of 0 and increasing it, but still having the vertex not pinned will **not** work (e.g. from goal = 0 to goal = 0.5) because the necessary "goal springs" cannot be generated on-the-fly.

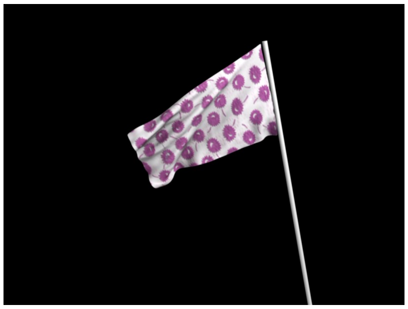
Using Cloth for Soft Bodies

Using cloth for soft bodies.

Cloth can also be used to simulate soft bodies. It is for sure not its main purpose but it works nonetheless. The example image uses standard *Rubber* material, no fancy settings, just `Alt-A`.

Blend-file for the example image: Using Cloth for soft bodies.

Cloth with Wind



Flag with wind applied.

Regression blend-file for Cloth with wind and self-collisions (also the blend for the image above): Cloth flag with wind and self-collisions.

Soft Body

- Introduction
  - Typical Scenarios for using Soft Bodies
  - Creating a Soft Body
  - Interaction in Real-Time
  - Tips
- Settings
  - Object
  - Simulation
  - Cache
  - Goal
  - Edges
  - Self Collision
  - Solver
- Forces
  - Exterior
  - Interior
- Collision
  - Collisions with Other Objects
  - Self-Collisions
- Examples
  - A Bouncing Cube

Introduction

Soft body simulation is used for simulating soft deformable objects. It was designed primarily for adding secondary motion to animation, like jiggle for body parts of a moving character.

It also works for simulating more general soft objects that bend, deform and react to forces like gravity and wind, or collide with other objects.

While it can simulate cloth and other stiff types of deformable objects to an extent, the Cloth Simulation can do it better with a solver specifically designed for this purpose.

The simulation works by combining existing animation on the object with forces acting on it. There are exterior forces like gravity or force fields and interior forces that hold the vertices together. This way you can simulate the shapes that an object would take on in reality if it had volume, was filled with something, and was acted on by real forces.

Soft bodies can interact with other objects through *Collision*. They can interact with themselves through *Self-Collision*.

The result of the soft body simulation can be converted to a static object. You can also *bake edit* the simulation, i.e. edit intermediate results and run the simulation from there.

Typical Scenarios for using Soft Bodies

The wind cone is a soft body, as the suspension.

Animation

Soft bodies are well suited for:

- Jiggle on moving characters.
- Elastic and deformable objects made of materials like rubber or gelatin.
- Tree branches moving in the wind, swinging ropes, and the like.
- Flags, wide sleeves, cushions or other simple fabric reacting to forces.

Creating a Soft Body

Soft body simulation works for all objects that have vertices or control points (meshes, curves, surfaces, and lattices).

To add a soft body simulation to an object, go to the *Physics* tab in the Properties and activate the *Soft Body* button. For a reference of all the settings see this page.

You start a soft body simulation by playback animation with `Alt-A`, and stop the simulation with `Esc` or `Alt-A`.

Interaction in Real-Time

To work with a soft body simulation, you will find it handy to use the Timeline editor. You can change between frames and the simulation will always be shown in the actual state. You can interact in real-time with the simulation, e.g. by moving collision objects or shaking a soft body object.

You can then select the soft body object while running the simulation and *Apply* the modifier in the *Modifiers* tab of the Properties. This makes the deformation permanent.

Tips

- Soft bodies work especially well if the objects have an even vertex distribution. You need enough vertices for good collisions. You change the deformation (the stiffness) if you add more vertices in a certain region.
- The calculation of collisions may take a long time. If something is not visible, why calculate it?
- To speed up the collision calculation it is often useful to collide with an additional, simpler, invisible, somewhat larger object.
- Use soft bodies only where it makes sense. If you try to cover a body mesh with a tight piece of cloth and animate solely with soft body, you will have no success. Self-collision of soft body hair may be activated, but that is a path that you have to wander alone. We will deal with Collisions in detail later.
- Try and use a *Lattice* or a *Curve Guide* soft body instead of the object itself. This may be magnitudes faster.

Settings

Reference

Panel:: *Physics* ▸ *Soft Body*

Collision Collection
: If set, soft body collides with objects from the collection, instead of using objects that are on the same layer.

- Object
- Simulation
- Cache
- Goal
    - Settings
    - Strengths
- Edges
    - Aerodynamics
    - Stiffness
- Self Collision
- Solver
    - Diagnostics
    - Helpers

Object

Friction

The friction of the surrounding medium. Generally friction dampens a movement. The larger the friction, the more viscous is the medium. Friction always appears when a vertex moves relative to its surround medium.

Mass

Mass value for vertices. Larger mass slows down acceleration, except for gravity where the motion is constant regardless of mass. Larger mass means larger inertia, so also braking a soft body is more difficult.

Control Point

You can paint weights and use a specified vertex group for mass values.

Simulation

Speed

You can control the internal timing of the soft body system with this value. It sets the correlation between frame rate and tempo of the simulation. A free falling body should cover a distance of about five meters after one second and travel at a speed of ten meters per seconds.

You can adjust the scale of your scene and simulation with this correlation. If you render with 25 frames per second, you will have to set *Speed* to 1.3.

Cache

Reference

Panel:: *Physics* ▸ *Soft Body* ▸ *Cache*

Soft Body physics simulations use a unified system for caching and baking. See Particle Cache and General Baking documentation for reference.

Goal

Reference

Panel:: *Physics* ‣ *Soft Body* ‣ *Goal*

Enabling this tells Blender to use the motion from animations (F-Curves, armatures, parents, lattices, etc.) in the simulation. The “goal” is the desired end position for vertices based on this animation.

See exterior forces for details.

Vertex Group
: Use a vertex group to allow per-vertex goal weights (multiplied by the *Default* goal).

Settings

Stiffness
: The spring stiffness for *Goal*. A low value creates very weak springs (more flexible “attachment” to the goal), a high value creates a strong spring (a stiffer “attachment” to the goal).

Damping
: The friction coefficient for *Goal*. Higher values give damping of the spring effect (little jiggle), and the movement will soon come to an end.

Strengths

Default
: Goal weight/strength for all vertices when no *Vertex Group* is assigned. If you use a vertex group the weight of a vertex defines its goal.

Min/Max

When you use a vertex group, you can use the *Minimum* and *Maximum* to fine-tune (clamp) the weight values. The lowest vertex weight will become *Minimum*, the highest value becomes *Maximum*.

Edges

Reference

Panel:: *Physics* ▸ *Soft Body* ▸ *Edges*

Allow the edges in a mesh object to act like springs. See interior forces.

Springs
: Use a specified vertex group for spring strength values.

Pull
: The spring stiffness for edges (how much the edges are allowed to stretch). A low value means very weak springs (a very elastic material), a high value is a strong spring (a stiffer material) that resists being pulled apart.

  A value of 0.5 is latex, 0.9 is like a sweater, 0.999 is a highly-starched napkin or leather. The soft body simulation tends to get unstable if you use a value of 0.999, so you should lower this value a bit if that happens.

Push
: How much the soft body resists being scrunched together, like a compression spring. Low values for fabric, high values for inflated objects and stiff material.

Damp
: The friction for edge springs. High values (max of 50) dampen the *Push*/*Pull* effect and calm down the cloth.

Plasticity
: Permanent deformation of the object after a collision. The vertices take a new position without applying the modifier.

Bending

This option creates virtual connections between a vertex and the vertices connected to its neighbors. This includes diagonal edges. Damping also applies to these connections.

Length

The edges can shrink or be blown up. This value is given in percent, 0 disables this function. 100% means no change, the body keeps 100% of its size.

Collision

Edge

Checks for edges of the soft body mesh colliding.

Face

Checks for any portion of the face of the soft body mesh colliding (which is computationally intensive). While *Face* enabled can solve collision errors, there does not seem to be any dampening settings for it. So parts of the soft body object near a collision mesh tend to “jitter” as they bounce off and fall back, even when there is no motion of any meshes. Edge collision has dampening, so that can be controlled, but Deflection dampening value on a collision object does not seem to affect the face collision.

Aerodynamics

Force from surrounding media. See exterior forces for details.

Type

Simple::	Edges receive a drag force from the surrounding media.
Lift Force::	Edges receive a lift force when passing through the surrounding media.

Factor

How much aerodynamic force to use. Try a value of 30 at first.

Stiffness

For quad faces, the diagonal edges are used as springs. This stops quad faces to collapse completely on collisions (what they would do otherwise).

Shear
  Stiffness of the virtual springs created for quad faces.

Self Collision

Reference

Panel::	*Physics ▸ Soft Body ▸ Self Collision*

Note

Self-Collision is working only if you have activated *Use Edges*.

When enabled, allows you to control how Blender will prevent the soft body from intersecting with itself. Every vertex is surrounded with an elastic virtual ball. Vertices may not penetrate the balls of other vertices. If you want a good result you may have to adjust the size of these balls. Normally it works pretty well with the default options.

Calculation Type

Manual::	The *Ball Size* directly sets the ball size.
Average::	The average length of all edges attached to the vertex is calculated and then multiplied with the *Ball Size* setting. Works well with evenly distributed vertices.
Minimal/Maximal::	The ball size is as large as the smallest/largest spring length of the vertex multiplied with the *Ball Size*.
Average Min Max::	Size = ((Min + Max)/2) × *Ball Size*.

Ball Size

Fraction of the length of attached edges. The edge length is computed based on the chosen algorithm. This setting is the factor that is multiplied by the spring length. It is a spherical distance (radius) within which, if another vertex of the same mesh enters, the vertex starts to deflect in order to avoid a self-collision.

Set this value to the fractional distance between vertices that you want them to have their own “space”. Too high of a value will include too many vertices at all times and slow down the calculation. Too low of a level will let other vertices get too close and thus possibly intersect because there will not be enough time to slow them down.

Stiffness

How elastic that ball of personal space is. A high stiffness means that the vertex reacts immediately to another vertex enters their space.

Dampening

How the vertex reacts. A low value just slows down the vertex as it gets too close. A high value repulses it.

Note

Collisions with other objects are set in the (other) Collision panel. To collide with another object they have to share at least one common layer.

Solver

Reference

Panel:: *Physics ‣ Soft Body ‣ Solver*

The settings in the *Soft Body Solver* panel determine the accuracy of the simulation.

Step Size Min
: Minimum simulation steps per frame. Increase this value, if the soft body misses fast-moving collision objects.

Max
: Maximum simulation steps per frame. Normally the number of simulation steps is set dynamically (with the *Error Limit*) but you have probably a good reason to change it.

Auto-Step
: Use velocities for automatic step sizes. Helps the Solver figure out how much work it needs to do based on how fast things are moving.

Error Limit
: Rules the overall quality of the solution delivered. The most critical setting that defines how precise the solver should check for collisions. Start with a value that is half the average edge length. If there are visible errors, jitter, or over-exaggerated responses, decrease the value. The solver keeps track of how “bad” it is doing and the *Error Limit* causes the solver to do some “adaptive step sizing”.

Diagnostics

Print Performance to Console
: Prints on the console how the solver is doing.

Estimate Transforms
    Estimate matrix, split to `COM`, `ROT`, `SCALE`.

Helpers

These settings control how the soft body will react (deform) once it either gets close to or actually intersects (cuts into) another collision object on the same layer.

Choke
    Calms down (reduces the exit velocity of) a vertex or edge once it penetrates a collision mesh.

Fuzzy
    Fuzziness while on collision, high values make collision handling faster but less stable. Simulation is faster, but less accurate.

Forces

- Exterior
  - Example
  - Force Fields
  - Aerodynamics
  - Goal
  - Technical Details
- Interior
  - Stiffness
  - Bending Stiffness

Exterior

Exterior forces are applied to the vertices (and nearly exclusively to the vertices) of soft body objects. This is done using Newton's Laws of Physics:

- If there is no force on a vertex, it stays either unmoved or moves with constant speed in a straight line.
- The acceleration of a vertex depends on its mass and the force. The heavier the mass of a vertex the slower the acceleration. The larger the force the greater the acceleration.
- For every action there is an equal and opposite reaction.

Well, this is done only in the range of computing accurateness, there is always a little damping to avoid overshoot of the calculation.

Example

We will begin with a very simple example: the default cube.

- To judge the effect of the external forces you should at first turn off the *Goal*, so that the vertices are not retracted to their original position.
- Start playback to run the simulation.

What happens? The cube moves in negative Z direction. Each of its eight vertices is affected by a global, constant force – the gravitation. Gravitation without friction is independent from the weight of an object, so each object you would use as a soft body here would fall with the same acceleration. The object does not deform, because every vertex moves with the same speed in the same direction.

Force Fields

Soft body vertices interact with all the Force Fields applied (usually to particles) in the layer, such as wind, force fields, and what ever physics field effect is on a common layer.

Soft Body Field Weights

Reference

Panel::	*Physics* ‣ *Soft Body* ‣ *Field Weights*

The *Soft Body Field Weights* panel allows you to control how much influence each type of external force field has on the soft body system.

Effector Collection
Limit effectors to a specified group. Only effectors in this group will have an effect on the current system.

Gravity
Control how much the Global Gravity has an effect on the system.

All
Scale all of the effector weights.

Aerodynamics

Edges can be affected by wind as they move, and sail or flutter in a breeze. A simple aerodynamic model of a flag sailing in the wind.

This special exterior force is not applied to the vertices but to the connecting edges. Technically, a force perpendicular to the edge is applied. The force scales with the projection of the relative speed on the edge (dot product). Note that the force is the same if wind is blowing or if you drag the edge through the air with the same speed. That means that an edge moving in its own direction subject to no force, and an edge moving perpendicular to its own direction is subjected to maximum force.

The angle and the relative speed between medium and edge is used to calculate the force on the edge. This force results that vertices with few connecting edges (front of a plane) fall faster than vertices with more connecting edges (middle of a plane). If all vertices have the same amount of edges in a direction they fall with equal speed.

The *Aerodynamics* settings are set in the Soft Body Edges panel.

Goal

A "goal" is a shape that a soft body object tries to conform to. It acts like a pin on a chosen set of vertices, controlling how much of an effect soft body has on them.

Enabling *Soft Body Goal* tells Blender to use the position (or animated position) of a vertex in the simulation. Animating the vertices can be done in all the usual ways (F-Curves, armatures, parents, lattices, etc.) before the soft body simulation is applied. The "goal" is the desired end position for vertices. How a soft body tries to achieve this goal can be defined using stiffness forces and damping.

See the Goal Settings for details.

Goal Strength

The *Goal Strength* defines how much motion from an animation system gets applied.

A *Goal* value of 1.0 means no soft body simulation, the object act like any regular animated object (i.e. the vertex is kept at its original position). When setting *Goal* to 0.0 (or no goal), the vertex is only influenced by physical laws according to soft body simulation.

By setting goal values between 0.0 and 1.0, you can blend between having the object affected only by the animation system, and having the object affected only by the soft body effect.

Goal also serves as a memory, to make sure soft objects don't deform too much, ending up in the non-soft animated shape. Using the *Vertex Group* weight system, you can define a *Goal* weight per vertex. To make this look more natural, spring forces can be defined to control how far vertices can move from their original position.

Often Weight Paint is used to adjust the weight comfortably. For non-mesh objects the *Weight* parameter of their vertices/control points is used instead; Use the Context menu in Edit Mode or the *Transform* panel in the Sidebar region. The weight of *Hair* particles can also be painted in Particle Edit Mode.

Technical Details

In the Soft Body world, vertices of meshes are treated as particles having a mass. Their movement is determined by the forces affecting them. Beside other forces the individual particles can interact with another along edges using a physical model which is very close to shock absorbers used in cars. The working parts are:

- A spring trying to keep the particles at a certain distance. How hard the spring tries to do that is controlled by the soft body parameter *Stiffness*.
- A damping element to calm the movement down. The resistance the element builds up against motion is controlled by the soft body parameter *Damping*.

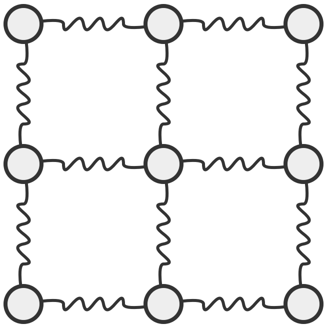
Interior

By default, the edges of a soft-body mesh act like springs. This means that, like a mechanical spring, they can stretch under tension and squeeze under pressure. Their initial length is also their “ideal” or “rest” length, which they try to return to.

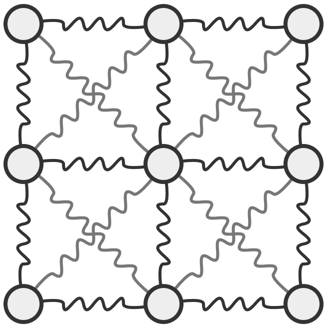
Having edges act like springs is what holds the mesh together. If you were to disable this behavior (as well as the Goal), each vertex would be free to go anywhere independently of the others, which would stretch the mesh until it’s no longer recognizable.

Having springs along edges alone typically isn’t enough, however: vertices in quads are still free to move towards their diagonal opposite, potentially collapsing the quad into a line.

You could solve this by creating diagonal edges everywhere, but fortunately, you don’t have to: simply enable the *Stiffness* option to have Blender create diagonal springs internally. This way, you don’t have to change your mesh.



Base springs along edges.



Additional springs when Stiffness is enabled.

Another method of preventing mesh collapse is applying *Bending Stiffness*, which adds rotational resistance: making edges try to keep their relative angles.

Both of these methods are described in more detail below. You can configure them, as well as other settings, in the Soft Body Edges panel.

Stiffness

To show the effect of the Stiffness setting, we will drop two cubes onto a plane (see Collisions). The blue cube uses quads, while the red one uses tris. Both cubes have their Goal setting disabled.

If *Stiffness* is disabled, the quad-only cube will collapse completely, while the tri cube only temporarily deforms from the impact:

Without Stiffness.

Frame 1.

Frame 36.

Frame 401.

If *Stiffness* is enabled, the quad cube maintains its shape as well thanks to the extra springs:

With Stiffness.

Frame 1.

Frame 36.

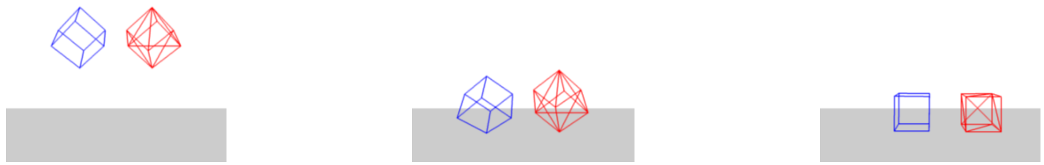
Frame 401.

Bending Stiffness

The second method to stop an object from collapsing is to give it *Bending Stiffness*. Just like the other settings, this can be combined with *Stiffness* to add bending resistance to the diagonal springs as well.

We first do the same cube experiment as before, using only *Bending Stiffness*:

Bending Stiffness.

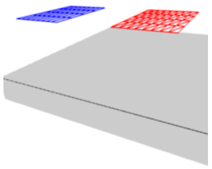


Frame 1.

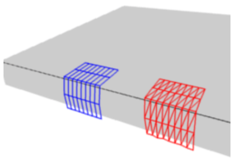
Frame 36.

Frame 401.

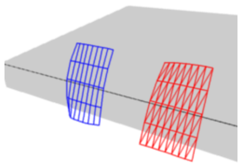
Both cubes keep their shape. Now, we try the same thing with subdivided planes, again a quad-based one and a triangulated one:



Two planes falling.



No bending stiffness.



High bending stiffness (10).

Without any *Bending Stiffness*, the faces can rotate freely as though their edges were hinges. Enabling *Stiffness* to add diagonal springs would not change this (just as triangulating doesn't).

With a high *Bending Stiffness*, however, the edges resist this rotation, and the planes act more like planks than towels.

Collision

There are two different collision types that you may use: collision between different objects and internal collision. We should set one thing straight from the start – the primary targets of the collision calculation are the vertices of a soft body. So if you have too few vertices too few collision takes place. Secondarily, you can use edges and faces to improve the collision calculation.

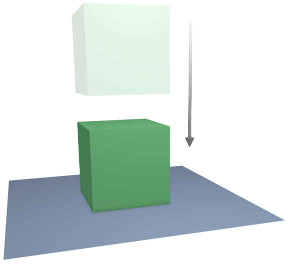
Collisions with Other Objects

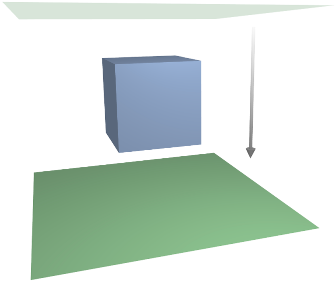
For a soft body to collide with another object there are a few prerequisites:

- If *Collision Collection* is set, the object must belong to the collection.
- The collision object has to be a mesh object.
- You have to activate the *Collision* in the *Physics* tab for the collision object. The collision object may also be a soft body.

Examples

A soft body cube colliding with a plane (Fig. A soft body cube colliding with a plane.) works pretty well, but a soft body plane falls right through a cube that it is supposed to collide with (Fig. A soft body plane colliding with a cube, so no interaction at all.).





A soft body cube colliding with a plane.	A soft body plane colliding with a cube, so no interaction at all.

Why is that? Because the default method of calculation only checks to see if the four vertices of the soft body plane collides with the cube as the plane is pulled down by gravity. You can activate *Collision: Face* (in the *Soft Body Edges* panel) to enable collision between the face of the plane and the object instead, but this type of calculation takes much longer.

Let us have a closer look at the collision calculation, so you can get an idea of how we might optimize it.

Calculating Collisions

Soft body simulations are by default done on a per-vertex basis. If the vertices of the soft body do not collide with the collision object, there will be no interaction between the two objects.

In the video below, you can see the vertices colliding with a plane. If a vertex penetrates the zone between *Outer* and *Inner*, it is repulsed by a force in the direction of the face normal. The position that a vertex finally ends up in is dependent on the forces that act upon it. In the example (the first vertex on the left in the video below) gravity and the repulsion force of the face balance out. The speed at which the vertex is pulled out of the collision zone is influenced by the *Choke* parameter in the Soft Body Solver settings.

See also

Download the blend-file.

Now let’s see what happens if we make vertices heavier and let them travel at a faster speed. In the video above, you can see vertices traveling at different speeds. The two on the far right (fifth and sixth) are traveling so fast that they pass right through the collision zone (this is because of the default solver precision, which we can fix later). You will notice that the fourth vertex also travels quite fast and because it is heavier it breaches the inner zone. The first three vertices collide correctly.

You can set up your collision so that edges and even faces are included in the collision calculation in the *Soft Body Edges* panel with the Collision *Face* and *Edge* options. The collision is then calculated differently. It is checked whether the edge or face intersects with the collision object, the collision zones are not used.

Good Collisions

If the collision you have set up is not behaving properly, you can try the following:

- The soft body object must have more subdivisions than the collision object. Add loop cuts to the soft body object in strategic areas that you know are most likely to be involved in a collision.
- Check the direction of the face normals.
- If the collision object has sharp spikes, they might penetrate the soft body.
- The resolution of the solver must match the speed at which soft body vertices are traveling. Lower the parameter *Error Limit* and carefully increase *Min Step*.
- *Outer* and *Inner* should be large enough, but zones of opposite faces should not overlap, or you have forces in opposite directions.
- If you use strong forces you should use large zones.
- Set *Choke* to a high enough value (all the way up if necessary) if you have difficulties with repelled vertices.
- Colliding faces are difficult to control and need long calculation times. Try not to use them.

Often it is better to create a simplified mesh to use as your collision object, however, this may be difficult if you are using an animated mesh.

Self-Collisions

For information on self-collision please refer to the Self Collision settings.

Examples

Here are some simple examples showing the power of soft body physics.

A Bouncing Cube

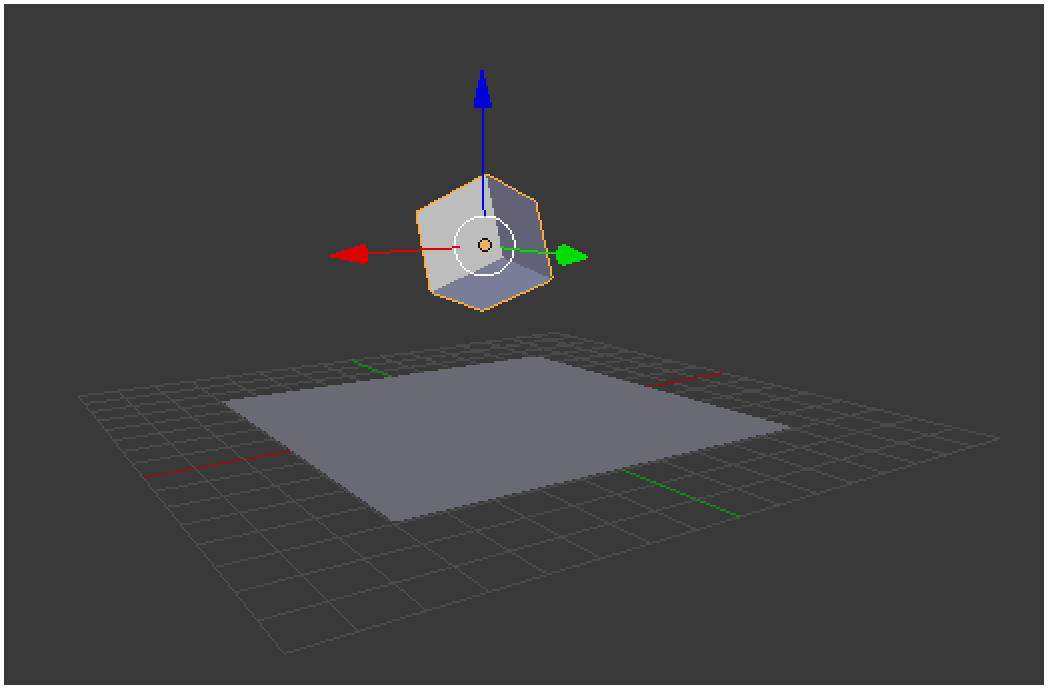
The Process

First, change your start and end frames to 1 and 150.

Then, add a plane, and scale it five times. Next, go to the physics tab, and add a collision. The default settings are fine for this example.

Now add a cube, or use the default cube, then enter *Edit Mode* to subdivide it three times. Add a Bevel Modifier to it to smoothen the edges and then to add a little more, press `R` twice, and move your cursor a bit.

When finished, your scene should look like this:



The scene, ready for soft body physics.

Everything is ready to add the soft body physics. Go to *Properties* ▸ *Physics* and choose *Soft Body*. Uncheck the *Soft Body Goal*, and check *Soft Body Self Collision*. Also, under *Soft Body Edges*, increase the Bending to 10.

Playing the animation will now give a slow animation of a bouncing cube. To speed things up, we need to bake the soft body physics.

Under *Soft Body Cache* change the values of your start and end frames. In this case 1 and 150. Now, to test if everything is working, you can take a cache step of 5 or 10, but for the final animation it is better to reduce it to 1, to cache everything.

You can now bake the simulation, give the cube materials and textures and render the animation.

The Result

The rendered bouncing cube

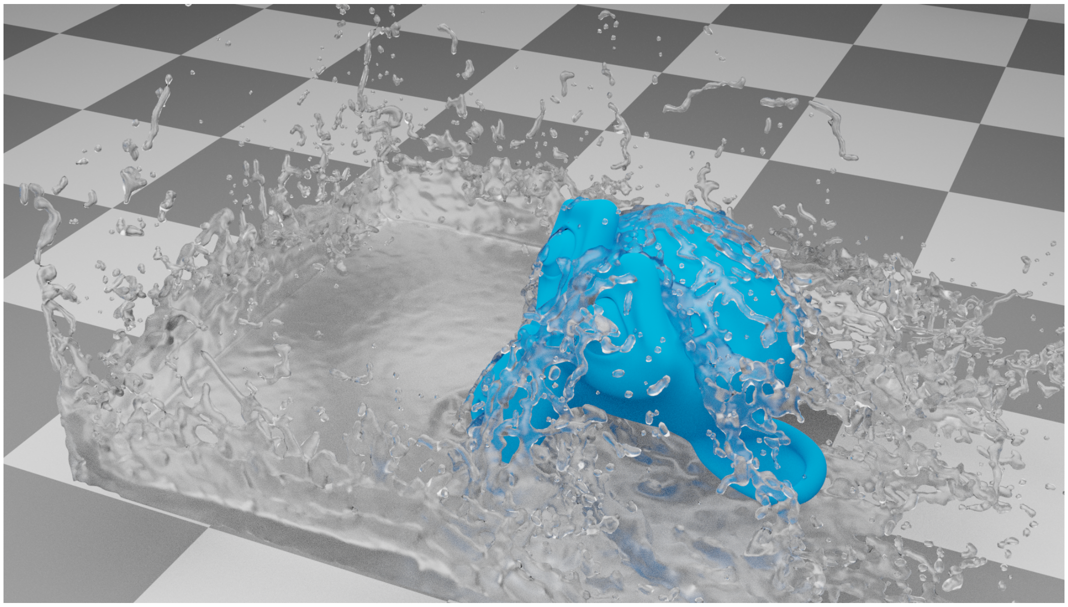
Fluid

- Introduction
    - Liquid Simulations
    - Gas Simulations
    - Workflow
- Type
    - Domain
    - Flow
    - Effector
- Materials
    - Smoke Material

Introduction

Liquid Simulations

Fluid physics are used to simulate physical properties of liquids especially water. While creating a scene in Blender, certain objects can be marked to become a part of the fluid simulation. For a fluid simulation you have to have a domain to define the space that the simulation takes up. In the domain settings you will be able to define the global simulation parameters (such as viscosity and gravity).



Example of a liquid simulation.

Gas Simulations

Gas or smoke simulations are a subset of the fluids system, and can be used for simulating collections of airborne solids, liquid particulates and gases, such as those that make up smoke. It simulates the fluid movement of air and generates animated Voxel textures representing the density, heat, and velocity of other fluids or suspended particles (e.g. smoke) which can be used for rendering.

Example of a fire simulation.

Gases or smoke are emitted inside of a Domain from a mesh object or particle system. The smoke movement is controlled by airflow inside the domain, which can be influenced by Effector objects. Smoke will also be affected by the scene's gravity and force fields. Airflow inside the domain can affect other physics simulations via the Fluid Flow force field.

Workflow

At least a Domain object and one Flow object are required to create a fluid simulation.

1. Create a Domain object that defines the bounds of the simulation volume.
2. Set up Flow objects which will emit fluid.
3. Set up Effector objects to make the fluid interact with objects in the scene.
4. Assign a material to the domain object.
5. Save the blend-file.

6. Bake the Cache for the simulation.

Note

There are Quick Liquid and Quick Smoke tools which will automatically create a domain object with a basic liquid or smoke and fire material.

Type

- Domain
  - Settings
  - Gas Settings
  - Liquid Settings
  - Guides
  - Collections
  - Cache
  - Field Weights
- Flow
  - Settings
- Effector
  - Settings

Domain

- Settings
    - Border Collisions
    - Gas
    - Fire
    - Liquid
- Gas Settings
    - Adaptive Domain
    - Noise
    - Viewport Display
- Liquid Settings
    - Diffusion
    - Particles
    - Mesh
- Guides
- Collections
- Cache
    - Advanced
- Field Weights

Settings

Reference

Panel::	*Physics* ‣ *Fluid* ‣ *Settings*
Type::	Domain

The domain object contains the entire simulation. Fluid simulations cannot leave the domain, it will either collide with the edge or disappear, depending on the domain's settings.

Keep in mind that large domains need higher resolutions and longer bake times. You will want to make it just large enough that the simulation will fit inside it, but not so large that it takes too long to compute the simulation.

To create a domain, add a cube and transform it until it encloses the area where you want the simulation to take place. Translation, rotation, and scaling are all allowed. To turn it into a fluid domain, click *Fluid* in the *Properties* ‣ *Physics* tab, then select *Domain* as the fluid *Type*.

Note

You *can* use other shapes of mesh objects as domain objects, but the fluid simulator will use the shape's Bounding Box as the domain bounds. In other words, the actual shape of the domain will still be rectangular.

Domain Type

A fluid domain can control either liquid or gas flows. Liquid domains take all liquid flow objects that intersect with the domain into consideration. Gas domains consider all intersecting *Smoke*, *Fire*, and *Smoke & Fire* flow objects. It is not possible to change the domain type dynamically.

Resolution Divisions

The fluid domain is subdivided into many “cells” called Voxels which make up “pixels” of fluid. This setting controls the number of subdivisions in the domain. Higher numbers of subdivisions are one way of creating higher resolution fluids.

Since the resolution is defined in terms of “subdivisions”, larger domains will need more divisions to get an equivalent resolution to a small domain. For example, a one meter cube with 64 *Resolution Divisions* will need 128 divisions to match a 2 meter cube. The dimension used as the base division is the longest dimension of the objects bounding box. To help visualize the voxel size, the *Resolution Divisions* can be previewed with a small cube shown in the 3D Viewport, to show the size of these divisions.

Time Scale

Controls the speed of the simulation. Low values result in a “slow motion” simulation, while higher values can be used to advance the simulation faster (good for generating fluids to be used in still renders).

Use Adaptive Time Steps

Lets the solver automatically decide when to perform multiple simulation steps per frame. It takes into account the maximum and minimum number of time steps, the current *Frame Rate*, and the *Time Scale*.

CFL Number

Determines the maximum velocity per grid cell and is measured in grid cells per time step. Fluid is only allowed to move up to this velocity in one time step. If this threshold is exceeded the solver will subdivide the simulation step.

In general, greater CFL (Courant–Friedrichs–Lewy) numbers will minimize the number of simulation steps and the computation time. Yet it will yield less physically accurate behavior for fast fluid flows. Smaller CFL numbers result in more simulation steps per frame, longer simulation times but more accurate behavior at high velocities (e.g. fast fluid flow colliding with obstacle).

Note

When lowering the *CFL* number it is recommended to increase the maximum number of time steps. Similarly, when increasing the *CFL* number the minimum number of time steps should be adjusted.

Timesteps Maximum
: Maximum number of allowed time steps per frame. If needed, the solver will divide a simulation step up to this number of sub-steps.

Timesteps Minimum
: Minimum number of allowed time steps per frame. The solver will always perform at least this number of simulation steps per frame.

Gravity
: By default the fluid solver will use the global scene gravity. This behavior can be disabled in the scene settings. Disabling the global gravity will enable the fluid gravity options.

Empty Space *Gas Only*
: Voxels with values under this value are considered empty space. More empty space optimizes rendering. With OpenVDB caching it also reduces cache sizes.

Delete in Obstacle
: Remover any volume of fluid that intersects with an obstacle inside the domain.

Border Collisions

Reference

Panel::	*Physics* ▸ *Fluid* ▸ *Settings* ▸ *Border Collisions*
Type::	Domain (Gas)

Controls which sides of the domain will allow fluid “pass through” the domain, making it disappear without influencing the rest of the simulation, and which sides will deflect fluids.

Gas

Reference

Panel::	*Physics* ‣ *Fluid* ‣ *Gas*
Type::	Domain (Gas)

Buoyancy Density

Buoyant force based on gas density.

- Values above 0 will cause the gas to rise (simulating gas which is lighter than ambient air).
- Values below 0 will cause gas to sink (simulating gas which is heavier than ambient air).

Buoyancy Heat

Controls how much gas is affected by temperature. The effect this setting has on gas depends on the per flow object Initial Temperature:

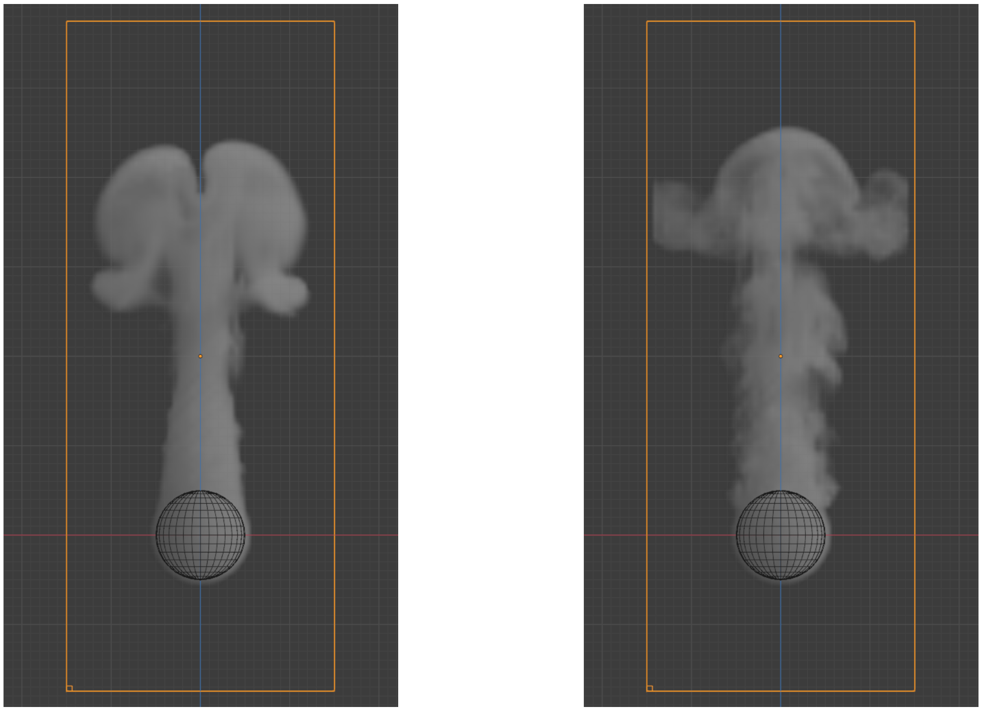
- Values above 0 will result in the gas rising when the flow object *Initial Temperature* is set to a positive value, and gas sinking when the flow object *Initial Temperature* is set to a negative value.
- Values below 0 will result in the opposite of positive values, i.e. gas emitted from flow objects with a positive *Initial Temperature* will sink, and gas from flow objects with a negative *Initial Temperature* will rise.

Note that gas from multiple flow objects with different temperatures will mix and warm up or cool down until an equilibrium is reached.

Vorticity

Controls the amount of turbulence in the gas. Higher values will make lots of small swirls, while lower values make smoother shapes.

Comparison of different amounts of vorticity.



Domain with a vorticity of 0.0.

Domain with a vorticity of 0.2.

Dissolve

Allow gas to dissipate over time.

Time

Speed of gas dissipation in frames.

Slow

Dissolve gas in a logarithmic fashion. Dissolves quickly at first, but lingers longer.

Fire

Reference

Type:: Domain
Panel:: *Physics* ▸ *Fluid* ▸ *Gas* ▸ *Fire*

Reaction Speed
How fast fuel burns. Larger values result in smaller flames (fuel burns before it can go very far), smaller values result in larger flames (fuel has time to flow farther before being fully consumed).

Flame Smoke
Amount of extra smoke created automatically to simulate burnt fuel. This smoke is best visible when using a “Fire + Smoke” Flow Object.

Vorticity
Vorticity for flames in addition to the global fluid Vorticity.

Temperature Maximum
Maximum temperature of flames. Larger values result in faster rising flames.

Minimum
Minimum temperature of flames. Larger values result in faster rising flames.

Smoke Color
Color of smoke emitted from burning fuel.

Liquid

Reference

Type:: Domain
Panel:: *Physics* ▸ *Fluid* ▸ *Liquid*

Liquid settings control the behavior of the particles which the simulation consists of. Enabling the liquid checkbox will automatically create a particle system for the simulation. This particle system visualizes the flow of the simulation. Visualizing the liquid particles is optional. The fluid simulation will make use of all the fields without an attached particle system too.

Note

Disabling the liquid checkbox will delete the attached particle system and its settings.

Simulation Method

Determines the liquid particle simulation method.

FLIP

Produces a very splashy simulation with lots of particles dispersed in the air.

APIC

Produces a very energetic but also more stable simulation. Vortices within the liquid will be preserved better than with *FLIP*.

FLIP Ratio *Simulation FLIP Only*:

How much FLIP velocity to use when updating liquid particle velocities. A value of 1.0 will result in a completely FLIP based simulation. Completely FLIP based simulations produce more chaotic splashes and are preferable when simulating greater quantities of liquid. When using smaller values the behavior will be less turbulent and splashes are more subtle. This is optimal when simulating scenes where the liquid is supposed to be on a small scale.

System Maximum

Maximum number of fluid particles that are allowed in the simulation. If this field is set to a nonzero value the simulation will never contain more than this number of fluid particles. Otherwise, with a value of zero the solver will always sample new particles when needed.

Particle Radius

The radius of one liquid particle in grid cells units. This value describes how much area is covered by a particle and thus determines how much area around it can be considered as liquid. A greater radius will let particles cover more area. This will result in more grids cell being tagged as liquid instead of just being empty.

Whenever the simulation appears to leak or gain volume in an undesired, non physically accurate way it is a good idea to adjust this value. That is, when liquid seems to disappear this value needs to be increased. The inverse applies when too much liquid is being produced.

Sampling

Factor that is used when sampling particles. A higher value will sample more particles. Note that particle resampling occurs at every single simulation step.

Randomness

New particles are sampled with some randomness attached to their position which can be controlled by this field. Higher values will sample the liquid particles more randomly in inflow regions. With a value of 0.0 all new particles will be sampled uniformly inside their corresponding grid cells.

When trying to create a laminar inflow (with little randomness) or more turbulent flows (with greater randomness) this value can be useful.

Particles Maximum

The maximum number of liquid particles per grid cell. During a simulation the number of liquid particles in a cell can fluctuate: Particles can flow into other cells or can get deleted if they move outside the narrow band. Resampling will add new particles considering this maximum.

This value sets the upper threshold of particles per cell. It is also a good way to estimate how many particles there can be in your simulation (one needs to take grid resolution into account too). This can be useful before baking and when planning a simulation.

Minimum

The minimum number of liquid particles per grid cell. Similarly to the maximum particle threshold, this value ensures that there are at least a certain amount of particles per cell.

Narrow Band Width

Controls the width in grid cell units of the narrow band that liquid particles are allowed to flow in. A high value will result in a thicker band and can result in an inflow region completely filled with particles. Unless the goal of the simulation is to visualize the liquid particles it is recommended to not increase the band width significantly as more particles slow down the simulation.

In some situations increasing this value can help create volume when the simulation appears to leak. In all other cases it is best to keep the narrow band as thin as possible since the liquid surface contains most details and simulating particles inside the liquid is not an optimal use of computing resources.

See also

The narrow band is an implementation of Narrow Band FLIP for Liquid Simulations.

Fractional Obstacles

Enables finer resolution in fluid / obstacle regions (second order obstacles). This option reduces the “stepping effect” that results when an obstacle lies inclined inside the domain. It also makes liquid flow more smoothly over an obstacle.

Obstacle Distance

Determines how far apart fluid and obstacles are. This value can be used to achieve a more fluid motion over inclined obstacles: Depending on the slope of the obstacle increasing this value can help liquid particles flow better over an obstacle. Setting this field to a negative value will let fluid move towards the inside of an obstacle.

Obstacle Threshold

Value to control the smoothness of the fractional obstacle option. Smaller value reduce the “stepping effect” but may result particles sticking to the obstacle.

Bake Data, Free Data

This option is only available when using the Modular cache type. *Bake Data* simulates and stores the base of the fluid simulation on drive. Both gas and liquid simulations can add refinements on top of this (e.g. gas simulations can add noise, liquid simulations can add a mesh or secondary particles or both).

The progress will be displayed in the status bar. Pressing `Esc` will pause the simulation.

Once the simulation has been baked, the cache can be deleted by pressing *Free Data*. It is possible to pause or resume a *Bake All* process.

Gas Settings

- Adaptive Domain
- Noise
- Viewport Display
  - Slice
  - Grid Display
  - Vector Display
  - Advanced *Gridlines Only*

Adaptive Domain

Reference

Type:: Domain
Panel:: *Physics ▸ Fluid ▸ Adaptive Domain*

When enabled, the domain will adaptively shrink to best fit the gas, saving computation time by leaving voxels without gas out of the simulation. Unless the *Add Resolution* is used, the adaptive domain will not exceed the bounds of the original domain.

Add Resolution
    Number of voxels to add around the outside of the domain.

Margin
    Amount of extra space to leave around gas, measured in voxels. With very fast-moving gas larger margins may be required to prevent the gas from being cut off by the adaptive boundary, but note this will increase the number of voxels which need to be computed.

Threshold
    Smallest amount of gas a voxel can contain before it is considered empty and the adaptive domain is allowed to cut it out of the simulation.

Noise

Reference

Type::	Domain
Panel::	*Physics* ▸ *Fluid* ▸ *Noise*

Adding noise to the gas simulation creates a finer detailed looking simulation on top of the base. This makes it possible to add more details to gases (i.e. fire or smoke or both) without changing the overall fluid motion.

See also

Fluid noise is an implementation of Wavelet Turbulence for Fluid Simulation.

Besides enabling parts of the interface, checking *Noise* lets the cache know which simulation data to read. If, for example, *Noise* is enabled but there is no noise simulation data to read it will show an empty domain. The checkbox does not reset the cache and can be used to switch the view between base resolution and noise view.

Upres Factor
: Factor by which to enhance the resolution of the noise. The scaling factor is coupled to the Resolution Divisions.

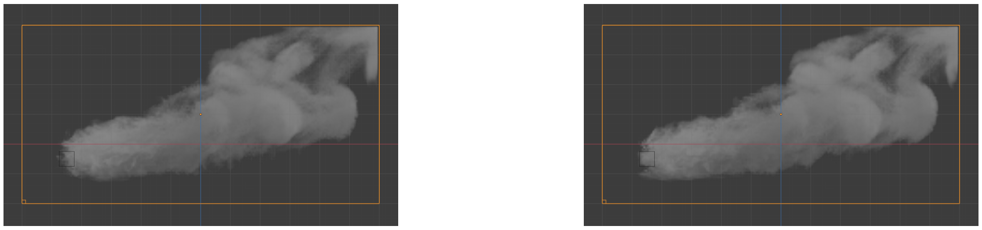
Strength
: Strength of the noise. Higher values result in more turbulent vortices.

Scale
: Scale of the noise. Greater values result in larger vortices.

Time

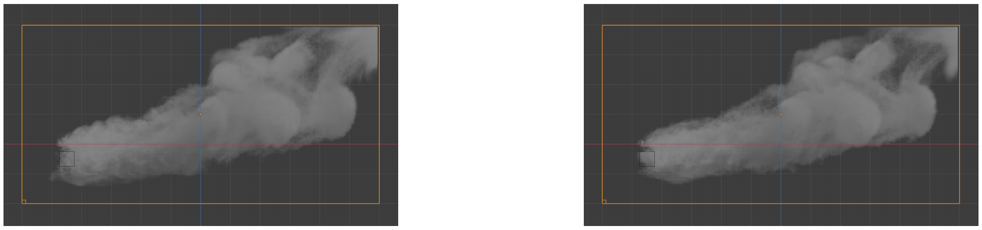
Animation time of the noise. This value has an influence on where the noise field is evaluated. It can be used as a seed to give wavelet noise a slightly different look in two domains that are otherwise the same.

Smoke plume with varying animation time. While the fluid motion of all four smoke plumes are alike each example has a unique look.



Animation Time: 0.1

Animation Time: 1.0



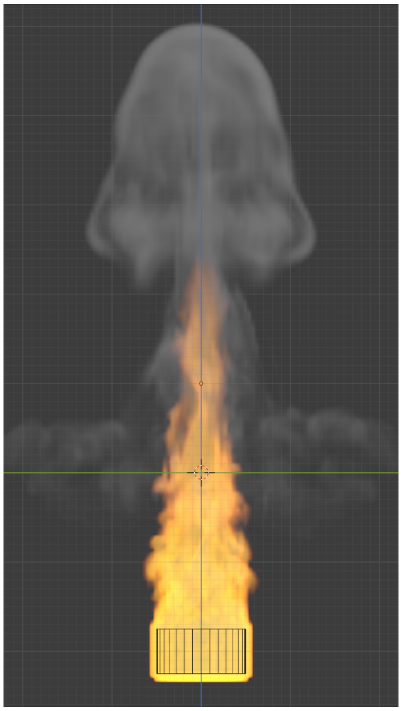
Animation Time: 2.5

Animation Time: 10.0

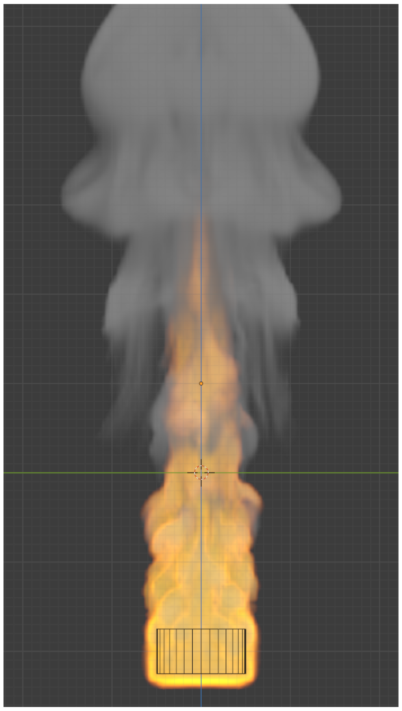
Note

Resolution Divisions and *Upres Factor* are not equivalent. By using different combinations of these resolution settings, you can obtain a variety of different styles of smoke.

Comparison of fire simulations with and without noise at the same grid resolution.



Resolution Divisions:
200, without noise



Resolution Divisions:
100, Noise scale: 2.

Low division simulations with lots of *Upres Factor* divisions generally appear smaller in real-world scale and can be used to achieve pyroclastic plumes such as in the following image:

Viewport Display

Thickness
  Factor that scales the thickness of the grid that is currently being displayed.

Interpolation
  Interpolation method to use for the visualization of the fluid grid.

  Linear
    Linear interpolation between voxels. Gives good smoothness and speed.

  Cubic
    Cubic interpolation between voxels. Gives smoothed high quality interpolation, but is slower.

  Closest
    No interpolation between voxels. Gives raw voxels.

Slice per Voxel
  Determines how many slices per voxel should be generated.

Slice

Renders only a single 2D section of the domain object.

Axis
  Auto
    Adjust slice direction according to the view direction.

  X/Y/Z
    Slice along the X/Y/Z axis.

Position

Position of the slice relative to the length of the respective domain side.

Gridlines *Closest Interpolation Only*

Display gridlines to differentiate the underlying cells in the current slice of the fluid domain.

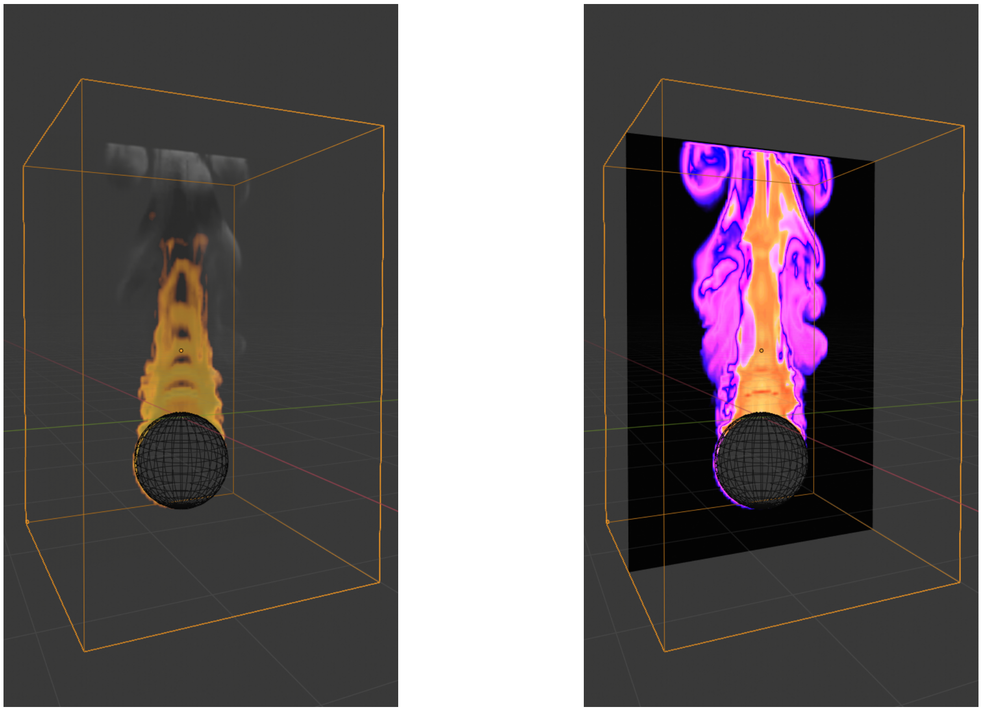
Grid Display

Use a specific color map for the visualization of the simulation field. This comes in handy during debugging or when making more advanced adjustments to the simulation. For instance, if the actual color of a fire simulation is barely visible in the viewport then changing the color profile can help to see the real size of the flame.

Field

The simulation field used in the display options (e.g. density, fuel, heat).

Comparison of a fire simulation with and without color mapping.



Slice view of “fire” grid without color mapping.

Slice view of “fire” grid with color mapping.

Scale
: Scale the selected simulation field by this value.

Vector Display

Visualization options for the vector fields.

Display As

- Streamlines
  - Choose to display the vectors as “Streamlines”.
- Needle
  - Choose to display the vectors as “Needles”.
- MAC Grid
  - Choose to display the vector field as “Marker-And-Cell Grid”.
  - X/Y/Z
    - Show an individual X/Y/Z component of the MAC grid.

Magnitude *Streamlines or Needle Only*

Scale the display vectors by the magnitude of the vectors they represent.

Field

The vector field represented by the display vectors (e.g. fluid velocity, external forces).

Scale

Scale the vectors by this size in the viewport.

Advanced *Gridlines Only*

Advanced coloring options for gridlines.

Color Gridlines

Flags

Color gridlines with flags.

Highlight Range *Grid Display Only*

Highlight the cells with values of the displayed grid within the range. Values between the *Lower Bound* and *Upper Bound* (inclusive) are considered to be within the range.

Lower Bound

Lower bound of the highlighting range.

Upper Bound

Upper bound of the highlighting range.

Color

Color used to highlight the cells.

Cell Type

Choose to highlight only a particular type of cells.

Liquid Settings

- Diffusion
  - High Viscosity Solver
- Particles
- Mesh

Diffusion

Reference

Type::	Domain
Panel::	*Physics ‣ Fluid ‣ Diffusion*

Liquid diffusion defines the physical properties of a liquid and in turn define how a liquid interacts with its environment. The main factors of diffusion are the *Viscosity* and *Surface Tension*. These properties can be adjusted to create virtual liquids that behave like water, oil, honey, or any other liquid. A couple presets exist to change the diffusion for different substances are predefined and can be changed in the preset menu. Fluid Diffusion settings can be enabled/disabled in the panel header.

Viscosity

> The viscosity refers to the “thickness” of the fluid and actually the force needed to move an object of a certain surface area through it at a certain speed.
>
> For manual entry, please note that real-world viscosity (the so-called dynamic viscosity) is normally measured in Pascal-seconds (\(Pa\cdot s\)), or in Poise units (P, equal to 0.1 \(Pa\cdot s\)), and commonly centiPoise units (cP, equal to 0.001 \(Pa\cdot s\)).
>
> Blender, on the other hand, uses the kinematic viscosity which is the dynamic viscosity divided by the density, \(\frac{Pa\cdot s}{kg/m^{3}}\), which is \(m^{2}/s\). So for example, the viscosity of water at room temperature is 1.002 cP, or 0.001002 \(Pa\cdot s\); the density of water is about 1000 \(kg/m^{3}\), which gives a kinematic viscosity of 0.000001002 \(m^{2}/s\) – so the entry would be 1.002 times 10 to the minus six (1.002×10^{-6} in scientific notation).
>
> The table below gives some examples of fluids together with their dynamic and kinematic viscosities.

Blender viscosity unit conversion.

Fluid	**Dynamic viscosity (in cP)**	**Kinematic viscosity (Blender, in \(m^{2}/s\))**
Water (20 °C)	1.002×10^{0} (1.002)	1.002×10^{-6} (0.000001002)
Oil SAE 50	5.0×10^{2} (500)	5.0×10^{-5} (0.00005)
Honey (20 °C)	1.0×10^{4} (10,000)	2.0×10^{-3} (0.002)
Chocolate Syrup	3.0×10^{4} (30,000)	3.0×10^{-3} (0.003)
Ketchup	1.0×10^{5} (100,000)	1.0×10^{-1} (0.1)
Melting Glass	1.0×10^{15}	1.0×10^{0} (1.0)

Tip

You can find the kinematic viscosity of more materials in the proper units by asking Wolfram Alpha, e.g. “kinematic viscosity of alcohol in m^2/s”.

To simplify the input of these numbers, the viscosity is changed by entering values in scientific notation by entering a base value and the exponent of that number.

Base

The base of the viscosity value (e.g. 1.002 in the case of water (20 °C)).

Exponent

The exponent of the viscosity value that will be multiplied by 10^{-1} (e.g. 6 in the case of water (20 °C)).

Note

Viscosity Varies

The default values in Blender are considered typical for those types of fluids and “look right” when animated. However, actual viscosity of some fluids, especially sugar-laden fluids like chocolate syrup and honey, depend highly on temperature and concentration. Oil viscosity varies by SAE rating. Glass at room temperature is basically a solid, but glass at 1500 °C flows (nearly) like water.

Warning

The simulator is not suitable for non-fluids, such as materials that do not “flow”. Simply setting the viscosity to very large values will not result in rigid body behavior, but might cause instabilities.

Surface Tension

Surface tension in grid units. Higher value will produce liquids with greater surface tension.

High Viscosity Solver

The high viscosity liquid solver can be used to simulate fluids with increased viscosity, replicating the behavior of substances like honey or molasses. This specialized solver enhances the accuracy of slow-moving and thick liquid simulations.

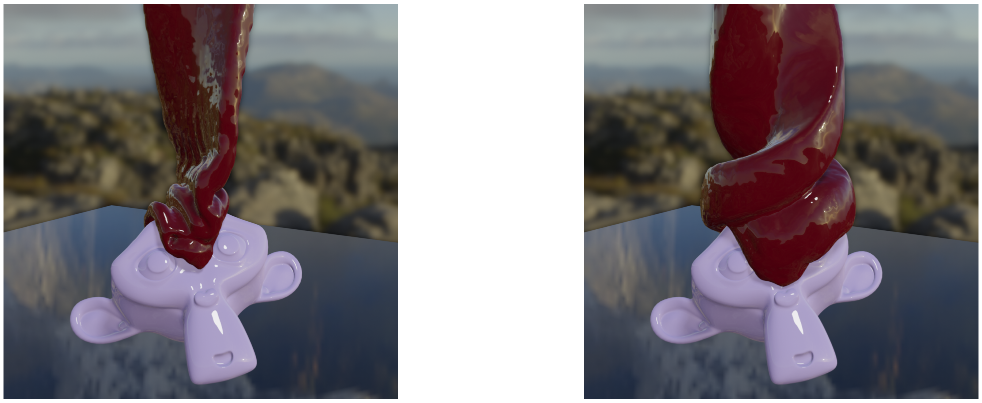
Strength

The viscosity of the liquid. Higher values result in more viscous fluids.

Note

A strength value of 0 will still apply some viscosity. Uncheck the *High Viscosity Solver* to disable the high viscosity liquid solver simulation step completely.

Rotating liquid inflow with varying viscosities.



Strength of 0.2 (at frame 65).

Strength of 0.4 (at frame 200).

Particles

Spray

Create spray particles during the secondary particle simulation. Spray particles are those that appear to fly through the air above the liquid surface when there is a bigger splash.

Foam

Create foam particles during the secondary particle simulation. Foam particles are those that solely move on the liquid surface.

Bubbles

Create bubble particles during the secondary particle simulation. Bubble particles are those that move below the liquid surface.

Note

Enabling a secondary particle type will also create a particle system for that type of particles. Disabling a particle type will delete this particle system including its settings.

Combined Export

Select particle types that should go into the same particle system. This option has no effect on the outcome of the simulation. It only changes the way particle systems are allocated in the particle settings.

Upres Factor

Factor by which to enhance the resolution of the particle simulation. The scaling factor is coupled to the Resolution Divisions (i.e. the particle simulation is this times bigger than the base simulation).

Wave Crest Potential Maximum

Upper clamping threshold for marking fluid cells as wave crests. A higher value results in less marked cells.

Wave Crest Potential Minimum

Lower clamping threshold for marking fluid cells as wave crests. A lower value results in more marked cells.

Trapped Air Potential Maximum

Upper clamping threshold for marking fluid cells where air is trapped. A higher value results in less marked cells.

Trapped Air Potential Minimum

Lower clamping threshold for marking fluid cells where air is trapped. A lower value results in more marked cells.

Kinetic Energy Potential Maximum

Upper clamping threshold that indicates the fluid speed where cells start to emit particles. A higher value results in generally less particles.

Kinetic Energy Potential Minimum

Lower clamping threshold that indicates the fluid speed where cells start to emit particles. A lower value results in generally more particles.

Potential Radius

Radius to compute potential for each cell. Higher values are slower but create smoother potential grids.

Particle Update Radius

Radius to compute position update for each particle. Higher values are slower but particles move less chaotic.

Wave Crest Particle Sampling

Maximum number of particles generated per wave crest cell per frame.

Trapped Air Particle Sampling

Maximum number of particles generated per trapped air cell per frame.

Particle Life Maximum

Highest possible particle lifetime.

Particle Life Minimum

Lowest possible particle lifetime.

Bubble Buoyancy

Amount of buoyancy force that rises bubbles. A high value results in bubble movement mainly upwards.

Bubble Drag

Amount of drag force that moves bubbles along with the fluid. A high value results in bubble movement mainly along with the fluid.

Particles in Boundary

Delete

Delete secondary particles that are inside obstacles or left the domain.

Push Out

Push secondary particles that left the domain back into the domain.

Bake Particles, Free Particles

This option is only available when using the Modular cache type.

The progress will be displayed in the status bar. Pressing `Esc` will pause the simulation.

Once the simulation has been baked, the cache can be deleted by pressing *Free Particles*. It is possible to pause or resume a *Bake Particles* process.

Mesh

The liquid mesh is, besides the liquid particles, another way to visualize the liquid simulation. It is generated directly from the liquid particles and often uses a higher resolution than the base Resolution Divisions.

Besides enabling parts of the interface, checking *Mesh* lets the cache know which simulation data to read. If, for example, *Mesh* is enabled but there is no mesh simulation data to read it will show the original domain. The checkbox does not reset the cache and can be used to switch the view between the original domain and the baked liquid mesh.

It is important to keep in mind that the shape of the mesh depends on a combination of *all* these parameters. E.g. changing the Particle Radius will lead to a different interpretation of the concavity values.

Upres Factor

> Factor by which to enhance the resolution of the mesh. The scaling factor is coupled to the Resolution Divisions (i.e. the mesh is this times bigger than the base simulation).

Particle Radius

> The radius of one liquid particle in grid cells units. This value describes how much area is covered by a particle and thus determines how much area around it can be considered as liquid. A greater radius will let particles cover more area. This will result in meshes covering more volume around liquid particles.
>
> This property refers to the same Particle Radius described in the liquid domain settings. Yet for the mesh, it is useful to interpret the particle radius on its own. For one, the mesh can have a resolution different from the base resolution through the Upres Factor. For another, it is often desirable to be able to control the mesh size around a single liquid particle.

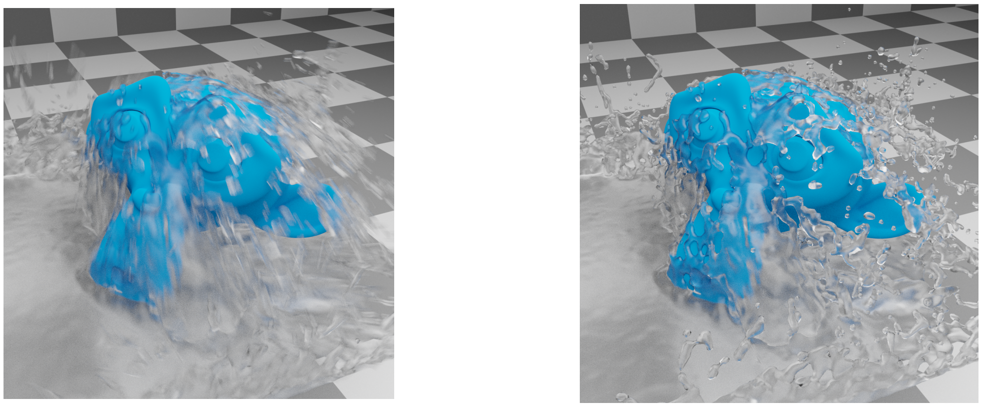
Use Speed Vectors

Creates a `velocity` Attribute which records the velocity of each vertex per frame. These will be used (automatically) when rendering with motion blur enabled.

Note

In order to render motion blur with Cycles, Deformation Motion Blur must be enabled.

Comparison of a liquid splash with and without motion blur (rendered with Cycles).



Motion blur enabled.

Motion blur disabled.

Mesh Generator

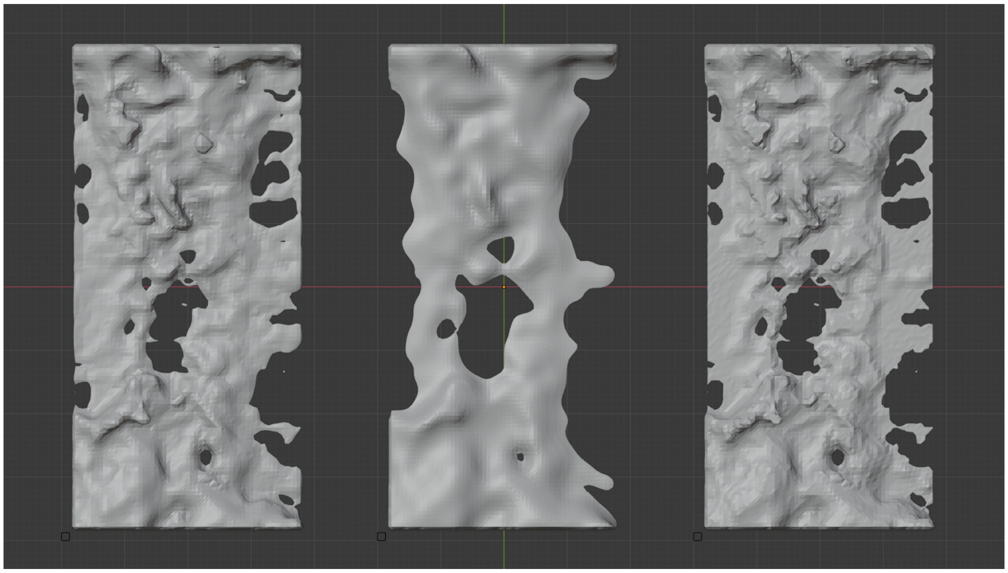
The mesh generator method determines the accuracy of the mesh. The *Final* option produces a higher quality mesh and provides more configuration option than the *Preview* option which in turn is faster but not as smooth.

Smoothing Positive

Positive mesh smoothing iterations. Higher values will make the mesh outline increasingly smooth. Yet higher values can prevent small details (e.g. smaller liquid drops) from getting meshed.

Smoothing Negative

Negative mesh smoothing iterations. Higher values will make the mesh outline sharper. High values will preserve details, however, the mesh outline will become more ragged (e.g. a single mesh particle will become less rounded and have more flat sides).



Comparison of a liquid drop hitting a surface (viewed from top) with varying smoothing values. Left: 1, 1 (Smoothing Positive, Smoothing Negative). Middle: 10, 1. Right: 1, 10. Note the slightly sharper corners in the right splash (compared to the left one).

Concavity Upper

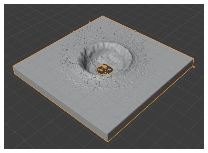
Upper mesh concavity bound. High values tend to smoothen and fill out concave regions.

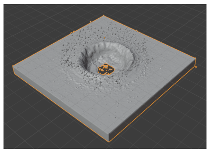
Concavity Lower

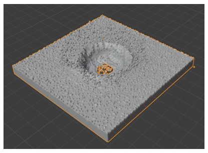
Lower mesh concavity bound. High values tend to smoothen and fill out concave regions.

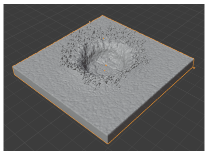
Using a lower concavity which is greater the upper concavity can result in distorted, non-manifold meshes. Unless the artist sees value in this kind of mesh, such concavity value combinations should be avoided.

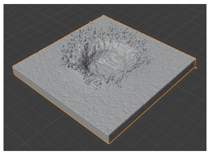
Crown splash with varying upper and lower concavity settings. Note that setting the concavity values to the same value produces a very granular mesh.

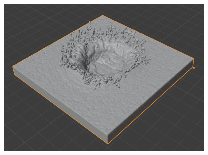

Upper: 1.0, Lower: 0.0.

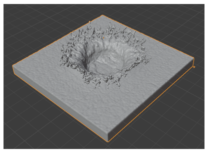

Upper: 1.0, Lower: 0.5.

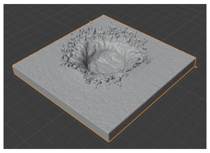

Upper: 1.0, Lower: 1.0.

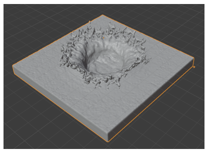

Upper: 1.5, Lower: 0.0.


Upper: 1.5, Lower: 0.5.


Upper: 1.5, Lower: 1.0.


Upper: 2.0, Lower: 0.0.


Upper: 2.0, Lower: 0.5.


Upper: 2.0, Lower: 1.0.

Bake Mesh, Free Mesh

This option is only available when using the Modular cache type.

The progress will be displayed in the status bar. Pressing `Esc` will abort the simulation.

Once the simulation has been baked, the cache can be deleted by pressing *Free Mesh*. It is possible to pause or resume a *Bake Mesh* process.

Guides

Reference

Panel::	*Physics* ▸ *Fluid* ▸ *Guides*
Type::	Domain

Fluid guides are used to apply forces onto the simulation. They are like simple external forces but also seek to preserve the physically accurate flow of the fluid. The *Guides* panel allows you to adjust guiding forces globally, i.e. for the entire domain. Enabling the guides hints the fluid solver to use the more accurate, but also computationally more expensive pressure solving step.

Even when there are no guiding objects baked or there is no guiding domain attached, the fluid solver will still perform the more expensive pressure guiding algorithm if guiding is enabled. It is therefore recommended to only enable *Guides* when there is a clear intention to use guiding in the simulation.

See also

Fluid guiding is an implementation of Primal-Dual Optimization for Fluids.

Weight
: Controls the lag of the guiding. A larger value (also known as the 'alpha' guiding value) results in a greater lag.

Size
: This setting determines the size of the vortices that the guiding produces. A greater guiding size (also known as the blur radius or 'beta' guiding value) results in larger vortices.

Velocity Factor

All guiding velocities are multiplied by this factor. That is, every cell of the guiding grid, which has the same size as the domain object, is multiplied by this factor.

Velocity Source

Guiding velocities can either come from objects that move inside the domain or from other fluid domains.

Effector

All effector objects inside the domain will be considered for the global guiding velocity grid. Once effector objects have been baked it is not possible to change the fluid domain resolution anymore.

Domain

When using another fluid domain as the guiding velocity source this domain may have a different resolution and may also be of a different type (e.g. the guiding domain is of type *Gas* while the actual domain with the guiding effect in it is of type *Liquid*).

In order to use a domain as the velocity source, this domain needs to be baked already.

Guide Parent

When using *Domain* as the velocity source, this field serves to select the guiding domain object.

Bake Guides, Free Guides

This option is only available when using the Modular cache type and when using *Effector* as the Velocity Source. *Bake Guides* writes vertex velocities of effector objects to drive. It is meant to be used before baking the fluid simulation.

The progress will be displayed in the status bar. Pressing `Esc` will pause the simulation.

Once the simulation has been baked, the cache can be deleted by pressing *Free Guides*. It is possible to pause or resume a *Bake Guides* process.

Collections

Reference

Type:: Domain
Panel:: *Properties* ▸ *Physics* ▸ *Fluid* ▸ *Collections*

Flow
: If set, only objects in the specified Collection will be allowed to act as Flow objects in this domain.

Effector
: If set, only objects in the specified Collection will be allowed to act as Effector objects in this domain.

Cache

Reference

Panel::	*Physics* ‣ *Fluid* ‣ *Cache*
Type::	Domain

The *Cache* panel is used to Bake the fluid simulation and stores the outcome of a simulation so it does not need to be recalculated.

Baking takes a **lot** of compute power (hence time). Depending on the scene, it is recommended to allocate enough time for the baking process.

If the mesh has modifiers, the rendering settings are used for exporting the mesh to the fluid solver. Depending on the setting, calculation times and memory use might exponentially increase. For example, when using a moving mesh with *Subdivision Surface* as an obstacle, it might help to decrease simulation time by switching it off, or to a low subdivision level. When the setup/rig is correct, you can always increase settings to yield a more realistic result.

Note

Fluid simulations use their own cache. All other physics simulations make use of the General Baking operators.

Cache Directory

Directory to store baked simulation files in. Inside this directory each simulation type (i.e. mesh, particles, noise) will have its own directory containing the simulation data.

Type

The type of the cache determines how the cache can be baked.

Replay

The cache will be baked as the simulation is being played in the viewport.

Modular

The cache will be baked step by step: The bake operators for this type are spread across various panels within the domain settings (e.g. the bake tool for the mesh can be found in the Mesh panel).

All

The cache will be baked with a single tool. All selected settings will be considered during this bake. The bake tool for this type can be found in the Cache panel.

Important

"Replay" only works when the Playback Sync mode is set to "Play Every Frame". If you need to use "Frame Dropping" or "Sync to Audio", consider using the "Modular" or "All" options below.

Start

Frame on which to start the simulation. This is the first frame that will be baked.

End

Frame on which to stop the simulation. This is the last frame that will be baked.

Note

The simulation is only calculated for positive frames between the *Start* and *End* frames of the *Cache* panel. So if you want a simulation that is longer than the default frame range you have to change the *End* frame.

Offset

Frame offset that is used when loading the simulation from the cache. It is not considered when baking the simulation, only when loading it.

Use Resumable Cache

Extra data will be saved so that you can resumed baking after pausing. Since more data will be written to drive it is recommended to avoid enabling this option when baking at high resolutions.

Volume File Format

File format for volume based simulation data (i.e. grids and particles).

Uni Cache

Blender's own caching format with some compression. Each simulation object is stored in its own `.uni` cache file.

OpenVDB

Advanced and efficient storage format. All simulation objects (i.e. grids and particles) are stored in a single `.vdb` file per frame.

Mesh File Format *Liquids Only*

File format for the mesh cache files.

Binary Object

Mesh data files with some compression.

Object

Simple, standard data format for mesh data.

Bake All, Free All

This option is only available when using the Final cache type. *Bake All* will run the simulation considering all parameters from the settings (i.e. it will bake all steps that can be baked individually with the Modular cache type at once).

The progress will be displayed in the status bar. Pressing `Esc` will abort the simulation.

Once the simulation has been baked, the cache can be deleted by pressing *Free All*. It is not possible to pause or resume a *Bake All* process as only the most essential cache files are stored on drive.

Advanced

Compression Volumes *OpenVDB Only*

Compression method that is used when writing OpenVDB cache files.

None

Cache files will be written without any compression.

Zip

Cache files will be written with `Zip` compression. Effective but slower than `Blosc`.

Blosc

Cache files will be written with `Blosc` compression. Multithreaded compression, similar in size and quality to `Zip` compression.

Precision Volumes *OpenVDB Only*

Precision level that is used when writing OpenVDB cache files.

Full

Volumetric data (e.g. grids, particles) will be written with full precision (32-bit).

Half

Volumetric data (e.g. grids, particles) will be written with half precision (16-bit).

Mini

Volumetric data (e.g. grids, particles) will be written with mini float precision (8-bit) where possible. For cache data where this is not possible, 16-bit floats will be used instead.

Export Mantaflow Script

Export the simulation as a standalone Mantaflow script when baking the scene (exported on “Bake Data”). Usually, only developers and advanced users who know how to use the Mantaflow GUI will make use of this functionality. Use a Debug Value of `3001` to enable.

Field Weights

Reference

Panel::	*Properties* ▸ *Physics* ▸ *Fluid* ▸ *Field Weights*
Type::	Domain

These settings determine how much gravity and Force Fields affect the fluid.

Effector Collection
: When set, fluid can only be influenced by force fields in the specified collection.

Gravity
: How much the fluid is affected by Gravity.

All
: Overall influence of all force fields.

The other settings determine how much influence individual force field types have.

Flow

Fluid *Flow* types are used to add or remove fluid to a domain object. Flow objects should be contained within the domain's Bounding Box in order to work.

To define any mesh object as a *Flow* object, add Fluid physics by clicking *Fluid* in *Properties ‣ Physics*. Then select *Flow* as the fluid *Type*. Now you should have a default fluid flow source object.

Settings

Reference

Panel::	*Physics ‣ Fluid ‣ Settings*
Type::	Flow

Flow Type

- Smoke
  - Emit only smoke.
- Fire + Smoke
  - Emit both fire and smoke.
- Fire
  - Emit only fire. Note that the domain will automatically create some smoke to simulate smoke left by burnt fuel.
- Liquid
  - Emit liquid.

Flow Behavior

Controls if the Flow object either adds (*Inflow*), removes (*Outflow*), or turns the mesh itself into fluid (*Geometry*).

Inflow

This object will emit fluid into the simulation, like a water tap or base of a fire.

Outflow

Any fluid that enters the Bounding Box of this object will be removed from the domain (think of a drain or a black hole). This can be useful in combination with an inflow to prevent the whole domain from filling up. Outflow objects can be animated and the area where the fluid disappears will follow the object as it moves around.

Geometry

All regions of this object that are inside the domain bounding box will be used as actual fluid in the simulation. You can place more than one fluid object inside the domain. Also make sure that the surface normals are pointing outwards or else they will not simulate properly. In contrast to domain objects, the actual mesh geometry is used for fluid objects.

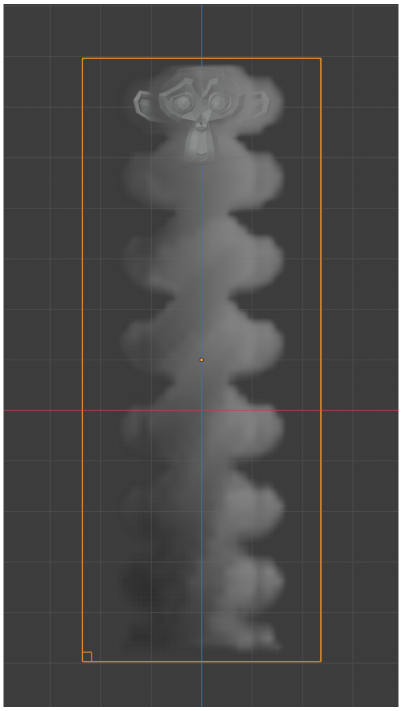
Use Flow

Enables or disables the flow of fluid, this property is useful for animations to selectively enable and disable when fluid is being added to or removed from the domain.

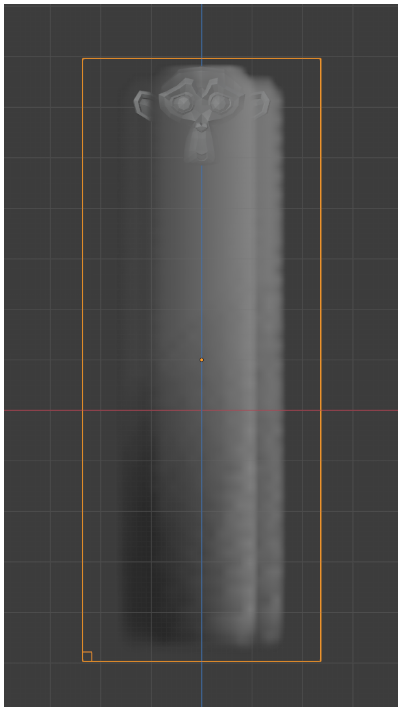
Sampling Substeps

Number of sub-steps used to reduce gaps in emission of fluid from fast-moving sources.

Comparison of smoke inflow quickly rising upwards at different sub-step rates.



Sampling sub-steps: 0.



Sampling sub-steps: 3.

Note that these emission sub-steps occur at every simulation step and not per frame. The simulation step count is controlled by the adaptive time stepping.

Smoke Color

The color of emitted smoke. When smoke of different colors are mixed they will blend together, eventually settling into a new combined color.

Absolute Density

If this checkbox is enabled, the emitter will only produce more smoke or fire if there is space for it in the emitter region. Otherwise smoke or fire will always be produced and add up.

Initial Temperature

Difference between the temperature of emitted smoke and the domain's ambient temperature. This setting's effect on smoke depends on the domain's Heat Buoyancy.

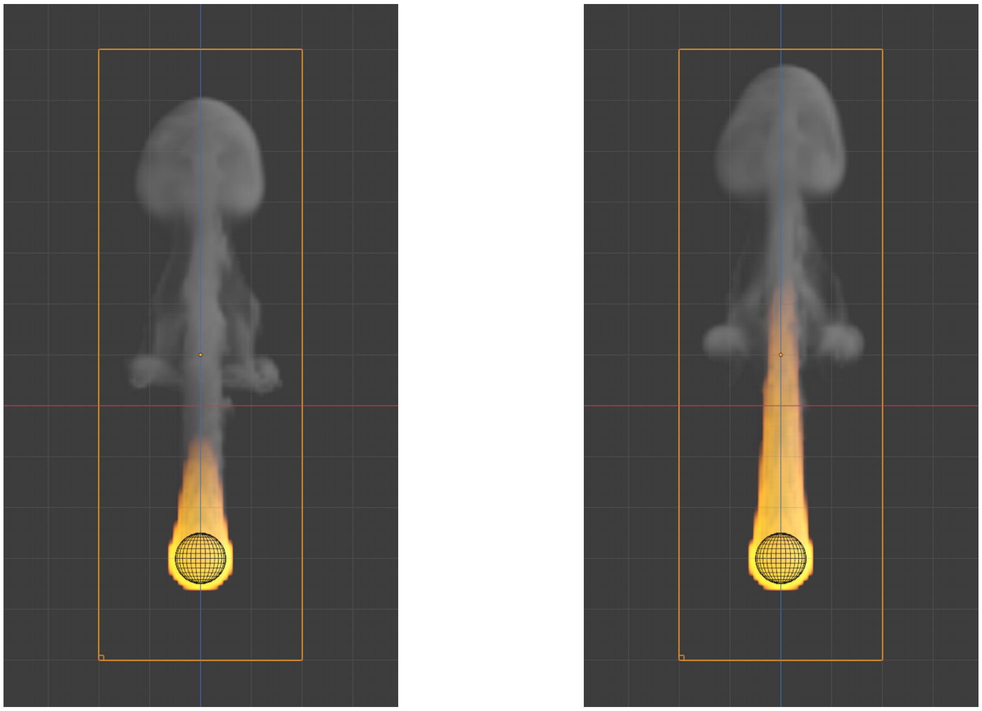
Density

Amount of smoke to emit at once. Larger values result in more density being produced.

Fuel

Amount of “fuel” being burned per second. Larger values result in larger flames, smaller values result in smaller flames:

Comparison of flames with varying fuel rates.



Fuel: 0.5.

Fuel: 1.0.

Vertex Group

When set, use the specified [Vertex Group](#) to control where smoke is emitted.

Flow Source

Flow Source

This setting defines the method used to emit fluid.

Mesh

Emit fluid directly from the object's mesh.

Is Planar

Defines the effector as either a single dimension object i.e. a plane or the mesh is Non-manifold. This ensures that the fluid simulator will give the most accurate results for these types of meshes.

Surface Emission

Maximum distance in Voxels from the surface of the mesh in which fluid is emitted. Since this setting uses voxels to determine the distance, results will vary depending on the domain's resolution.

Volume Emission *Fire or Smoke Only*:

Amount of fluid to emit inside the emitter mesh, where 0 is none and 1 is the full amount. Note that emitting fluid based on volume can have unpredictable results if your mesh is Non-manifold.

Particle System *Fire or Smoke Only*:

Create smoke or fire from a particle system on the flow object. The particle system can be selected with a Data ID.

Note that only *Emitter* type particle systems can add smoke. See Particles for information on how to create a particle system.

Set Size

When this setting is enabled, it allows the *Size* setting to define the maximum distance in voxels at which particles can emit smoke, similar to the *Surface Emission* setting for mesh sources.

When disabled, particles will fill the nearest Voxel with smoke.

Initial Velocity

When enabled, the fluid will inherit the momentum of the flow source.

Source
: Factor for the inherited velocity. A value of 1 will emit fluid moving at the same speed as the source.

Normal
: This option controls how much velocity fluid is given along a face Normal. Note that, initial velocities will always be applied along all face normals. Thus with a closed flow source mesh, fluid will always be emitted in more than one direction. To set initial velocities along only one direction all normals need to point in the same direction. This is can be achieved when using a plane as the flow object.

Initial X, Y, Z
: Initial velocity along X, Y, Z coordinates in world space. This can be used in addition to the initial velocity along the Normal.

Texture

Reference

Type::	Flow
Panel::	*Physics ‣ Fluid ‣ Settings ‣ Texture*

When enabled, use the specified texture and settings to control where on the mesh smoke or fire can be emitted from. These settings have no effect on *Outflow Flow Behavior*.

Texture
: A Data ID selector to choose the Texture.

Mapping
: Controls whether to use Generated UVs or manual UV mapping.

Size
: Overall texture scale.

Offset

Effector

Effector objects are used to deflect fluids and influence the fluid flow. To define any mesh object as an effector object, add fluid physics by clicking *Fluid* in *Properties* ‣ *Physics*. Then select *Effector* as the fluid *Type*.

Tip

Force Fields (such as wind or vortex) are supported, like in most physics systems. The influence individual force types have can be controlled per domain object.

Settings

Reference

Panel::	*Physics* ‣ *Fluid* ‣ *Settings*
Type::	Effector

Effector Type

Collision

Objects of this type will collide with fluid.

Guide

The velocity of objects of this type will be used when baking the guiding. So fluid guiding objects should move and have some velocity.

Velocity Factor

Multiply the guiding object velocities by this factor. This is useful when working with multiple guiding objects and some of them should have higher or smaller velocities.

Guide Mode

The mode describes how guiding velocities should be written into the global guiding velocity field of the domain.

Maximize

The guiding object will compare the existing velocity in the global velocity field with its own velocity. If its absolute value is greater than the absolute value in the velocity field the guiding velocity will be kept.

Minimize

A guiding object will compare the existing velocity in the global velocity field with its own velocity. If its absolute value is smaller than the absolute value in the velocity field the guiding velocity will be kept.

Override

The most intuitive option. A guiding object will always write its own current velocity into the global guiding velocity field. Values in the velocity field from a previous frame or guiding object will be overridden.

Averaged

A guiding object will write the average of its own current velocity and the existing guiding velocity at that cell into the global guiding velocity field.

Effector Substeps

Number of substeps used to reduce gaps in collision of fluid from fast-moving effectors.

Surface Thickness

Additional area around the effector that will be considered as an effector.

Use Effector

Enables or disables the effector object effect on the fluid, this property is useful for animations to selectively enable and disable when the effector affects the fluid.

Is Planar

Defines the effector as either a single dimension object i.e. a plane or the mesh is Non-manifold. This ensures that the fluid simulator will give the most accurate results for these types of meshes.

A Manifold mesh can also be declared as planar. The fluid solver will then ignore the volume inside the mesh and just emit fluid from the mesh sides.

Materials

Smoke Material

Realistic smoke can be rendered with the Principled Volume shader.

Smoke Material Example Animation

Particle System

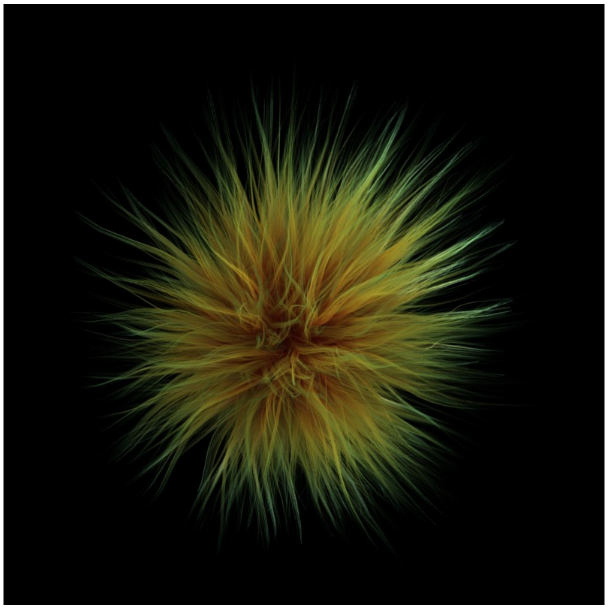
- Introduction
- Particle System Panel
  - Workflow
- Emitter
  - Emission
  - Cache
  - Velocity
  - Rotation
  - Physics
  - Render
  - Viewport Display
  - Children
  - Force Fields
  - Vertex Groups
- Hair
  - Introduction
  - Emission
  - Hair Dynamics
  - Render
  - Shape
  - Children
  - Viewport Display
- Texture Influence
  - General
  - Physics
  - Hair
- Particle Edit Mode
  - Usage
  - Selecting
  - Tools
  - Options
  - Editing

Introduction

Particles are lots of items emitted from mesh objects, typically in the thousands. Each particle can be a point of light or a mesh, and be joined or dynamic. They may react to many different influences and forces, and have the notion of a lifespan. Dynamic particles can represent fire, smoke, mist, and other things such as dust or magic spells.

Hair type particles are a subset of regular particles. Hair systems form curves that can represent hair, fur, grass and bristles.

You see particles as a Particle Modifier, but all settings are done in the *Particle tab*.



Some fur made from particles.

Particles generally flow out from their mesh into space. Their movement can be affected by many things, including:

- Initial velocity out from the mesh.
- Movement of the emitter (vertex, face or object) itself.

- Movement according to “gravity” or “air resistance”.
- Influence of force fields like wind, vortexes or guided along a curve.
- Interaction with other objects like collisions.
- Partially intelligent members of a flock (herd, school, …), that react to other members of their flock, while trying to reach a target or avoid predators.
- Smooth motion with soft body physics (only *Hair* particle systems).
- Or even manual transformation with Lattices.

Particles may be rendered as:

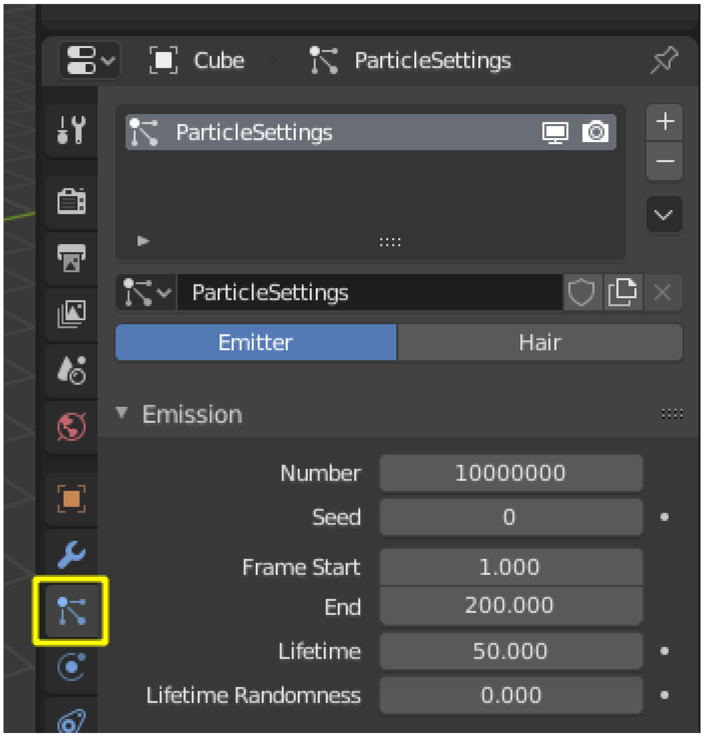
- Halos (for Flames, Smoke, Clouds).
- Meshes which in turn may be animated (e.g. fish, bees, …). In these cases, each particle “carries” another object.
- Hair curves, following the path of the particle. These hair curves can be manipulated in the 3D Viewport (combing, adding, cutting, moving, etc.).

Every object may carry many particle systems. Each particle system may contain up to 10,000,000 particles. Certain particle types (*Hair* and *Keyed*) may have up to 10,000 children for each particle (children move and emit more or less like their respective parents). The size of your memory and your patience are your practical boundaries.

Particle System Panel

Reference

Panel:: *Particle System* ▸ *Particle System*



Cube

ParticleSettings

Emitter

Hair

Emission

Number: 10000000

Seed: 0

Frame Start: 1.000

End: 200.000

Lifetime: 50.000

Lifetime Randomness: 0.000

Particle System panel.

These are the basic settings.

Active Particle System

The List View of the objects Particle Modifier(s).

Specials

Copy Active to Selected Objects

Copies the active particle system to all selected objects.

Copy All to Selected Objects

Copies all particle systems from the active object to all selected objects.

Duplicate Particle Systems

Duplicates the particle system within the active object. The *Duplicate Settings* option (in the Adjust Last Operation panel) will duplicate settings as well, so the new particle system uses its own settings.

Particle Settings

The Data-Block menu for settings.

Type

Main selector of the system type.

Emitter

In such a system, particles are emitted from the object.

Hair

Use Hair type, rendered as strands.

Regrow

Regrows the hair for each frame. This is useful when you are animating properties.

Advanced

Enables advanced settings which reflect the same ones as working in Emitter mode.

Note

This manual assumes that this option is enabled.

Segments

Controls the number of parts a hair is made of. Increasing this value will improve the quality of animations.

Workflow

The process for working with standard particles is:

1. Create the mesh which will emit the particles.
2. Create one or more Particle Systems to emit from the mesh. Many times, multiple particle systems interact or merge with each other to achieve the overall desired effect.
3. Tailor each Particle System's settings to achieve the desired effect.
4. Animate the base mesh and other particle meshes involved in the scene.
5. Define and shape the path and flow of the particles.
6. For Hair particle systems: Sculpt the emitter's flow (cut the hair to length and comb it for example).
7. Make final render and do physics simulation(s), and tweak as needed.

Creating a Particle System

To add a new particle system to an object, go to the *Particles* tab of the Properties editor and click the small + button. An object can have many Particle Systems.

Each particle system has separate settings attached to it. These settings can be shared among different particle systems, so one does not have to copy every setting manually and can use the same effect on multiple objects.

Types of Particle Systems

After you have created a particle system, the Properties fills with many panels and buttons. But do not panic! There are two different types of particle systems, and you can change between these two with the *Type* selector: Emitter and Hair.

The settings in the *Particle System* tab are partially different for each system type.

Emitter

- Emission
  - Source
- Cache
  - Hints
- Velocity
- Rotation
  - Angular Velocity
- Physics
  - Introduction
  - Newtonian
  - Keyed
  - Boids
  - Fluid
- Render
  - Common Settings
  - Render As
  - Extra
- Viewport Display
- Children
  - Common Options
  - Simple
  - Interpolated
  - Example
- Force Fields
  - Field Weights
  - Force Fields Settings
- Vertex Groups

Emission

Reference

Panel:: *Particle System ‣ Emission*

The *Emitter* system works just like its name says: it emits/produces particles for a certain amount of time. In such a system, particles are emitted from the selected object from the *Start* frame to the *End* frame and have a certain lifespan. These particles are rendered default as Halos, but you may also render this kind of particles as objects (depending on the particle system's render settings, see Visualization).

The buttons in the *Emission* panel control the way particles are emitted over time:

Number
: The maximum amount of parent particles used in the simulation.

Seed
: Blender uses this as starting point to produce random numbers during the simulation.

Frame Start
: The start frame of particle emission. You may set negative values, which enables you to start the simulation before the actual rendering.

End
: The end frame of particle emission.

Lifetime
: The lifespan (in frames) of the particles.

Lifetime Randomness

A random variation of the lifetime of a given particle. The shortest possible lifetime is *Lifetime* × (1 - *Random*). Values above 1.0 are not allowed. For example with the default *Lifetime* value of 50 a *Random* setting of 0.5 will give you particles with a live span ranging from 50 frames to $50 \times (1.0 - 0.5) = 25$ frames, and with a *Random* setting of 0.75 you will get particles with live spans ranging from 50 frames to $50 \times (1.0 - 0.75) = 12.5$ frames.

Source

Reference

Panel:: *Particle System* ▸ *Emission* ▸ *Source*

Emit From

Defines how and where the particles are emitted, giving precise control over their distribution.

> **Tip**
>
> You may use vertex groups to confine the emission, that is done in the *Vertex Groups* panel.

Vertices

Emits particles from the vertices of a mesh.

Faces

Emits particles from the surface of a mesh's faces.

Volume

Emits particles from the volume of an enclosed mesh.

> **Tip**
>
> Your mesh must be Manifold to emit particles from the volume. Some modifiers like the Edge Split Modifier break up the surface,

in which case volume emission will not work correctly!

Use Modifier Stack

Take any Modifiers above the Particle Modifier in the modifier stack into account when emitting particles, else it uses the original mesh geometry.

Note

Note that particles may differ in the final render if these modifiers generate different geometry between the viewport and render.

Distribution

These settings control how the emissions of particles are distributed throughout the emission locations when emitting from either *Faces* or *Volume*.

Jittered

Particles are placed at jittered intervals on the emitter elements.

Particles/Face

Number of emissions per face (0 = automatic).

Jittering Amount

Amount of jitter applied to the sampling.

Random

Particles are emitted from random locations in the emitter's elements.

Grid

Particles are set in a 3D grid and particles near/in the elements are kept.

Invert Grid

Invert what is considered the object and what is not.

Hexagonal

Uses a hexagonal-shaped grid instead of a rectangular one.

Resolution

Resolution of the grid.

Random

Add a random offset to grid locations.

Random Order

The emitter element indices are gone through in a random order instead of linearly (one after the other).

Not available for *Grid* distribution.

Even Distribution

Particle distribution is made even based on surface area of the elements, i.e. small elements emit less particles than large elements, so that the particle density is even.

Cache

Reference

Panel:: *Particle System ‣ Cache*

In order to improve real-time response and avoid unnecessary recalculation of particles, the particle data can be cached in memory or stored on a drive.

The *Emitter* particle system uses a unified system for caching and baking (together with Soft Body and Cloth).

See also

See the General Baking docs for more information.

Hints

- The simulation is only calculated for positive frames in between the *Start* and *End* frames of the *Cache* panel, whether you bake or not. So if you want a simulation that is longer than the default frame range, you have to change the *End* frame.
- When an animation is played, each physics system writes each frame to the cache. Note that for the cache to fill up, one has to start the playback before or on the frame that the simulation starts.
- The cache is cleared automatically on changes. But not on all changes, so it may be necessary to delete it manually, e.g. if you change a force field.
- The system is protected against changes after baking. If for example the mesh changes the simulation is not calculated anew.
- The bake result can be cleared by clicking on the *Delete Bake* button in the simulation cache settings.

- A simulation can only be edited in Particle Edit Mode when it has been baked in memory. And cannot be edited if the Disk Cache is used.
- If you are not allowed to write to the required subdirectory caching will not take place, e.g. if your blend-file path is very long and your operating system has a limit on the path length that is supported.
- Be careful with the sequence of modifiers in the modifier stack. You may have a different number of faces in the 3D Viewport and for rendering (e.g. when using subdivision surface), if so, the rendered result may be very different from what you see in the 3D Viewport.

Velocity

Reference

Panel:: *Particle System ‣ Velocity*

The initial velocity of particles can be set through different parameters, based on the type of the particle system. If the particle system type is Emitter or Hair, then the following parameters give the particle an initial velocity.

Normal
    The emitter's surface normals (i.e. let the surface normal give the particle a starting speed).

Tangent
    Let the tangent speed give the particle a starting speed.

Tangent Phase
    Rotates the surface tangent.

Object Align
    Give an initial velocity in the X, Y, and Z axes.

    X, Y, Z

Object Velocity
    The emitter objects movement (i.e. let the object give the particle a starting speed).

Randomize
    Gives the starting speed a random variation. You can use a texture to only change the value, see Controlling Emission, Interaction and Time.

Rotation

Reference

Panel:: *Particle System ‣ Rotation*

These parameters specify how the individual particles are rotated at the start of, and during, their lifetime. You can visualize their orientation by setting *Display As* to *Axis* in the Viewport Display panel.

Orientation Axis

Aligns the X axis of new particles to:

None

The global X axis.

Normal

The emitter’s surface normal.

Normal-Tangent

The emitter’s surface normal, additionally aligning the particle’s Y axis to the positive V direction in the emitter’s active UV map. This makes it possible to deform the emitter while keeping particle rotation consistent.

Velocity / Hair

The particle’s initial velocity vector/hair growth direction.

Global X, Y, Z

One of the global axes.

Object X, Y, Z

One of the emitter’s local axes.

Randomize

How much to randomize the particle’s initial rotation (along all axes).

Phase

Initial rotation around the particle’s X axis, going from -1 (-180°) to 1 (180°).

Randomize Phase

Maximum random rotation to add to the *Phase*, going from 0 (0°) to 2 (360°).

Dynamic

Whether the particles’ rotation can change over time.

Angular Velocity

Reference

Panel:: *Particle System ‣ Rotation ‣ Angular Velocity*

Lets you configure if and how particles should spin over time. *Dynamic* needs to be enabled for this to work.

Axis

The axis to spin around. If this is set to *Velocity*, *Horizontal*, or *Vertical*, particles will additionally spin to keep the same orientation relative to their direction of movement, even if *Amount* is zero.

None

Spinning is disabled.

Velocity

Spin around the particle’s velocity vector.

Horizontal

Spin around the axis that’s horizontal (lying in the global XY plane) and perpendicular to the particle’s velocity. Particles moving along the global Z axis won't spin because no unique rotation axis exists in this case.

Vertical

Spin around the axis that's perpendicular to both the particle's velocity and the above *Horizontal* axis. Particles moving along the global Z axis won't spin.

Global X, Y, Z

Spin around the chosen global axis.

Random

Spin around a random axis.

Hint

If you use a Curve Guide and want the particles to always point in the direction of the curve, you should set the *Orientation Axis* to *Velocity / Hair*, enable *Dynamic*, and set the *Angular Velocity Axis* to *Velocity*.

(For a regular object, you'd normally use the *Follow Curve* option of a Follow Path Constraint or the legacy Follow option of the curve itself, but these don't work for particles.)

Amount

How fast to spin around the *Axis*.

Physics

- Introduction
    - Common Physics Settings
    - No Physics
- Newtonian
    - Forces
    - Integration
    - Deflection
- Keyed
    - Options
    - Relations
- Boids
    - Movement
    - Battle
    - Misc
    - Relations
    - Boid Brain
- Fluid
    - Options

Introduction

The movement of particles may be controlled in a multitude of ways. Here we will discuss only the particle physics in the narrower sense, i.e. the settings in the Physics panel.

Additional ways of moving particles are:

- By soft body animation (only for Hair particle systems).
- By force fields and along curves.
- By lattices.

Common Physics Settings

Size
Sets the size of the particles.

Random Size
Give the particles a random size variation.

Mass
Specify the mass of the particles.

Multiply Mass with Particle Size
Causes larger particles to have larger masses.

No Physics

The particles will be given no motion, which makes them belong to no physics system. At first a physics type that makes the particles to be static could seem a bit strange, but it can be very useful at times. None physics make the particles stick to their emitter their whole life time. The initial velocities here are for example used to give a velocity to particles that are affected by a harmonic effector with this physics type when the effect of the effector ends.

Moreover, it can be very convenient to have particles at disposal (whose both Unborn and Died are visible on render) to groom vegetation and/or ecosystems using *Object* or *Group* types of visualization.

Newtonian

Reference

Panel::	*Particle System* ‣ *Physics*
Type::	Newtonian

The particles will move according to classical (Newtonian) mechanics. Particles start their life with the specified initial velocities and angular velocities, and move according to external forces. The response to environment and to forces is computed differently, according to the given integrator chosen by the animator.

Forces

Reference

Panel::	*Particle System* ‣ *Physics* ‣ *Forces*

Brownian
: Specify the amount of Brownian motion. Brownian motion adds random motion to the particles based on a Brownian noise field. This is nice to simulate small, random wind forces.

Drag
: A force that reduces particle velocity in relation to its speed and size (useful in order to simulate air drag or water drag).

Damp
: Reduces particle velocity (deceleration, friction, dampening).

Integration

Reference

Panel:: *Particle System ‣ Physics ‣ Integration*

Integrators are a set of mathematical methods available to calculate the movement of particles. The following guidelines will help to choose a proper integrator, according to the behavior aimed at by the animator.

Integration

Euler

Also known as “Forward Euler”. Simplest integrator. Very fast but also with less exact results. If no dampening is used, particles get more and more energy over time. For example, bouncing particles will bounce higher and higher each time. Should not be confused with “Backward Euler” (not implemented) which has the opposite feature, the energy decrease over time, even with no dampening. Use this integrator for short simulations or simulations with a lot of dampening where speedy calculations are more important than accuracy.

Verlet

Very fast and stable integrator, energy is conserved over time with very little numerical dissipation.

Midpoint

Also known as “2nd order Runge-Kutta”. Slower than Euler but much more stable. If the acceleration is constant (no drag for example), it is energy conservative. It should be noted that in example of the bouncing particles, the particles might bounce higher than they started once in a while, but this is not a trend. This integrator is a generally good integrator for use in most cases.

RK4

Short for “4th order Runge-Kutta”. Similar to Midpoint but slower and in most cases more accurate. It is energy conservative even if the acceleration is not constant. Only needed in complex simulations where Midpoint is found not to be accurate enough.

Timestep
  The amount of simulation time (in seconds) that passes during each frame.

Subframes
  The number of simulation steps per frame. Subframes to simulate for improved stability and finer granularity in simulations. Use higher values for faster-moving particles.

The following options are only available for Fluid type physics:

Adaptive
  Automatically set the number of subframes.

  Threshold
    A tolerance value that allows the number of subframes to vary. It sets the relative distance a particle can move before requiring more subframes.

    The number of steps per frame will be at least Subframes + 1. More subframes may be simulated if the fluid becomes turbulent, according to the Threshold.

Deflection

Reference

Panel::	*Particle System ‣ Physics ‣ Deflection*

Size Deflect
  Use the particle size in deflections.

Die on Hit
  Kill particle when it hits a deflector object.

Collision Collection
  If set, particles collide with objects from the collection.

Keyed

Reference

Panel::	*Particle System ‣ Physics*
Type::	Keyed

The path of Keyed particles is determined between particles of any two (or more) particle systems. This allows the creation of a chains of systems to create long strands or groovy moving particles. Basically the particles have no dynamics but are interpolated from one system to the next each frame.

To setup *Keyed* particles you need at least two particle systems in the *Keys* list.

Options

Loops
: Sets the number of times the entire *Keys* list is repeated. Disabled if *Use Timing* is enabled.

Use Timing
: Enabling this option allows you to specify the timing for each key independently, using the *Time* and *Duration* options. By default, the *Use Timing* option is deactivated, and the particles will pass through all keys for a time equal to its lifetime. A shorter lifetime means faster movement. The lifetime will be split equally between the keys, this may lead to varying particle speeds between the targets.

Relations

Reference

Panel:: *Particle System ▸ Physics ▸ Relations*

Key Targets
The list view of keys (target particle systems).

Object
The name of a target object for the selected key. If empty it uses the current particle system.

System
Index of particle system on the target object.

Time
The time (frame number) at which the particles will be at the position of the selected system. Note also that the *Start* frame of the Keyed system adds an offset to this time.

Duration
How long (in frames) the particles stay on this system before they start moving to the next one.

Boids

Reference

Panel::	*Particle System* ‣ *Physics*
Type::	Boids

Boids particle systems are controlled by a limited artificial intelligence, which can be programmed to follow basic rules and behaviors. They are ideal for simulating flocks, swarms, herds and schools of various kind of animals, insects and fishes or predators vs. preys simulations. They can react on the presence of other objects and on the members of their own system. Boids can handle only a certain amount of information, therefore the sequence of the *Boid Brain* rules is very important. In certain situations only the first three parameter are evaluated.

Movement

Reference

Panel::	*Particle System* ‣ *Physics* ‣ *Movement*

Boids try to avoid objects with activated Collision. They try to reach goal objects, and fly from “predators” according to the *Boid Brain* settings.

Boids can have different physics depending on whether they are in the air, or on land (on collision object).

Allow Flight
    Allow boids to move in the air.

Allow Land
    Allow boids to move on land.

Allow Climbing
Allow boids to climb goal objects.

Max Air Speed
Set the Maximum velocity in the air.

Min Air Speed
Set the Minimum velocity in the air.

Max Air Acceleration
Lateral acceleration in air, percentage of the max velocity (turn). Defines how fast a boid is able to change direction.

Max Air Angular Velocity
Tangential acceleration in air, percent 180 degrees. Defines how much the boid can suddenly accelerate in order to fulfill a rule.

Air Personal Space
Radius of boids personal space in air. Percentage of particle size.

Landing Smoothness
How smoothly the boids land.

Max Land Speed
Set the Maximum velocity on land.

Jump Speed
Maximum speed for jumping.

Max Land Acceleration
Lateral acceleration on land, percent of max velocity (turn). Defines how fast a boid is able to change direction.

Max Land Angular Velocity
Tangential acceleration on land, percent 180 degrees. Defines how much the boid can suddenly accelerate in order to fulfill a rule.

Land Personal Space
Radius of boids personal space on land. Percentage of particle size.

Land Stick Force
How strong a force must be to start effecting a boid on land.

Collision Collection
Only collide with objects in this collection.

Battle

Reference

Panel:: *Particle System ‣ Physics ‣ Battle*

Health
Initial boid health when born.

Strength
Maximum caused damage per second on attack.

Aggression
Boid will fight this time stronger than enemy.

Accuracy
Accuracy of attack.

Range
Maximum distance of which a boid can attack.

Misc

Reference

Panel:: *Particle System ‣ Physics ‣ Misc*

Banking
Amount of rotation around velocity vector on turns. Banking of 1.0 gives a natural banking effect.

Pitch
Amount of rotation around side vector.

Height
Boid height relative to particle size.

Relations

Reference

Panel:: *Particle System* ▸ *Physics* ▸ *Relations*

Target
This list view allows you to set up other particle systems to react with the boids.

Target Object
A data ID to select an object with a particle system set on.

System
Index of the *Object*‘s particle system as set in the list view in the particle panel.

Mode
  Enemy
    Setting the type to *Enemy* will cause the systems to fight with each other.

  Friend
    Will make the systems work together.

  Neutral
    Will not cause them to align or fight with each other.

Deflection

Boids will try to avoid deflector objects according to the Collision rule's weight. It works best for convex surfaces (some work needed for concave surfaces).

Force Fields

As other physics types, Boids is also influenced by external force fields.

In addition, special *Boid* force fields can be used with the Boids physics. These effectors could be predators (positive Strength) that boids try to avoid, or targets (negative Strength) that boids try to reach according to the (respectively) *Avoid* and *Goal* rules of the *Boid Brain*.

Boid Brain

Reference

Panel::	*Particle System ‣ Physics ‣ Boid Brain*

The Boid Brain panel controls how the boids particles will react with each other. The boids' behavior is controlled by a list of rules. Only a certain amount of information in the list can be evaluated. If the memory capacity is exceeded, the remaining rules are ignored.

The rules are by default parsed from top-list to bottom-list (thus giving explicit priorities), and the order can be modified using the little arrows buttons on the right side.

Rule Evaluation
  There are three ways to control how rules are evaluated:

  Average
    All rules are averaged.

  Random
    A random rule is selected for each boid.

Fuzzy

Uses fuzzy logic to evaluate rules. Rules are gone through top to bottom. Only the first rule that affect above the *Rule Fuzziness* threshold is evaluated. The value should be considered how hard the boid will try to respect a given rule (a value of 1 means the Boid will always stick to it, a value of 0 means it will never). If the boid meets more than one conflicting condition at the same time, it will try to fulfill all the rules according to the respective weight of each.

Note

A given boid will try as much as it can to comply to each of the rules it is given, but it is more than likely that some rule will take precedence on other in some cases. For example, in order to avoid a predator, a boid could probably "forget" about Collision, Separate and Flock rules, meaning that "while panicked" it could well run into obstacles, e.g. even if instructed not to, most of the time.

In Air

The current rule affects boids while they are flying.

On Land

The current rule affects boids while they are not flying.

Goal Rule

Seek goal.

Object

Specifies the goal object. If not specified, Boid force fields with negative Strength are used as goals.

Predict

Predict target's movements.

Avoid Rule

Avoid “predators”.

Object
: Specifies the object to avoid. If not specified, Boid force fields with positive Strength are used as predators.

Predict
: Predict target’s movements.

Fear Factor
: Avoid object if danger from it is above this threshold.

Avoid Collision Rule

Avoid objects with activated Deflection.

Boids
: Avoid collision with other boids.

Deflectors
: Avoid collision with deflector objects.

Look Ahead
: Time to look ahead in seconds.

Separate Rule

Boids move away from each other.

Flock Rule

Copy movements of neighboring boids, but avoid each other.

Follow Leader Rule

Follows a leader object instead of a boid.

Distance
    Distance behind leader to follow.

Line
    Follow the leader in a line.

    Queue Size
        How many boids that are allowed to follow in a line.

Average Speed Rule

Maintain average velocity.

Speed
    Percentage of maximum speed.

Wander
    How fast velocity's direction is randomized.

Level
    How much velocity's Z component is kept constant.

Fight Rule

Move toward nearby boids.

Fight Distance
    Attack boids at a maximum of this distance.

Flee Distance
    Flee to this distance.

Fluid

Reference

Panel::	*Particle System ▸ Physics*
Type::	Fluid

Fluid particles are similar to Newtonian ones but this time particles are influenced by internal forces like pressure, surface tension, viscosity, springs, etc. From liquids to slime, goo to sand and wispy smoke the number of possible use cases is endless.

Blender particle fluids use the SPH techniques to solve the particles fluid equations. Smoothed-particle hydrodynamics (SPH) is a computational method used for simulating fluid flows. It has been used in many fields of research, including astrophysics, ballistics, vulcanology, and oceanography. It is a mesh-free Lagrangian method (where the coordinates move with the fluid), and the resolution of the method can easily be adjusted with respect to variables such as the density.

Options

Fluid physics share options with Newtonian Physics. These are covered on that page.

Fluid Properties

Stiffness
  How incompressible the fluid is.

Viscosity
  Linear viscosity. Use lower viscosity for thicker fluids.

Buoyancy

Artificial buoyancy force in negative gravity direction based on pressure differences inside the fluid.

Advanced

Reference

Panel:: *Particle System ‣ Physics ‣ Advanced*

Repulsion Factor

How strongly the fluid tries to keep from clustering (factor of stiffness). Checkbox sets repulsion as a factor of stiffness.

Stiff Viscosity

Creates viscosity for expanding fluid. Checkbox sets this to be a factor of normal viscosity.

Interaction Radius

Fluid's interaction radius. Checkbox sets this to be a factor of $4 \times \textit{particle size}$.

Rest Density

Density of fluid when at rest. Checkbox sets this to be a factor of default density.

Springs

Reference

Panel:: *Particle System ‣ Physics ‣ Springs*

Force

Spring force.

Rest Length

Rest length of springs. Factor of particle radius. Checkbox sets this to be a factor of $2 \times \textit{particle size}$.

Viscoelastic Springs

Use viscoelastic springs instead of Hooke’s springs.

Elastic Limit

How much the spring has to be stretched/compressed in order to change its rest length.

Plasticity

How much the spring rest length can change after the elastic limit is crossed.

Initial Rest Length

Use initial length as spring rest length instead of 2 × *particle size*.

Frames

Create springs for this number of frames since particle’s birth (0 is always).

Render

Reference

Panel:: *Particle System ‣ Render*

The Render Panel controls how particles appear when they are rendered.

Note

Cycles supports only Object and Collection render types.

Common Settings

Scale
Todo.

Scale Randomness
Todo.

Material
Set which of the object's materials is used to shade the particles.

Coordinates System
Use a different object's coordinates to determine the birth of particles.

Show Emitter
When disabled, the emitter is no longer rendered. Activate the button *Emitter* to also render the mesh.

Render As

None

When set to *None*, particles are not rendered. This is useful if you are using the particles to duplicate objects.

Halo

Halos are rendered as glowing dots or a little cloud of light. Although they are not really lights because they do not cast light into the scene like a light object. They are called *Halos* because you can see them, but they do not have any substance.

Path

The *Path* visualization needs a Hair particle system or Keyed particles.

B-Spline
: Interpolate hair using B-splines. This may be an option for you if you want to use low *Render* values. You loose a bit of control but gain smoother paths.

Steps
: Set the number of subdivisions of the rendered paths (the value is a power of 2). You should set this value carefully, because if you increase the render value by two you need four times more memory to render. Also the rendering is faster if you use low render values (sometimes drastically). But how low you can go with this value depends on the waviness of the hair (the value is a power of 2). This means 0 steps give 1 subdivision, 1 give 2 subdivisions, $2 \rightarrow 4, 3 \rightarrow 8, 4 \rightarrow 16, \ldots\ n \rightarrow 2^n$.

Timing

Reference

Panel::	*Particle System* ▸ *Render* ▸ *Timing*
Type::	Hair

Absolute Path Time
Path timing is in absolute frames.

End
End time of the practical path.

Random
Give the path length a random variation.

Object

Reference

Panel:: *Particle System ‣ Render ‣ Object*

Instance Object
The specified object is instanced in place of each particle.

Global Coordinates
Use object's global coordinates for instancing.

Object Rotation
Use the rotation of the object.

Object Scale
Use the size of the object.

Collection

Reference

Panel:: *Particle System ‣ Render ‣ Collection*

Instance Collection
The objects that belong to a collection are instanced sequentially in the place of the particles.

Whole Collection
Use the whole group at once, instead of one of its elements, the group being displayed in place of each particle.

Pick Random
The objects in the group are selected in a random order, and only one object is displayed in place of a particle. Please note that this mechanism fully replaces old Blender particles system using parentage and *Instancing Vertices* to replace particles with actual geometry. This method is fully deprecated and does not work anymore.

Global Coordinates
Use object's global coordinates for instancing.

Object Rotation
Use the rotation of the objects.

Object Scale
Use the size of the objects.

Use Count

Reference

Panel::	*Particle System ‣ Render ‣ Collection ‣ Use Count*

Use objects multiple times in the same groups. Specify the order and number of times to repeat each object with the list view that appears. You can duplicate an object in the list with the + button, or remove a duplicate with the – button.

Extra

Reference

Panel::	*Particle System ‣ Render ‣ Extra*

Parents Particles

Render also parent particles if child particles are used. Children have a lot of different deformation options, so the straight parents would stand between their curly children. So by default *Parents* are not rendered if you activate *Children*. See Children.

Unborn

Render particles before they are born.

Dead

Render particles after they have died. This is very useful if particles die in a collision *Die on hit*, so you can cover objects with particles.

Viewport Display

Reference

Panel:: *Particle System ‣ Viewport Display*

The Display Panel controls how particles are displayed in the 3D Viewport. This does not necessarily determine how they will appear when rendered.

Display As

- None
  - The particles are not shown in the 3D Viewport and are not rendered. The emitter may be rendered though.
- Rendered
  - Particles are displayed the way they are rendered.
- Point
  - Particles are displayed as square points. Their size is independent of the distance from the camera.
- Circle
  - Particles are displayed as circles that face the view. Their size is independent of the distance from the camera.
- Cross
  - Particles are displayed as 6-point crosses that align to the rotation of the particles. Their size is independent of the distance from the camera.
- Axis
  - Particles are displayed as 3-point axes. This is useful if you want to see the orientation and rotation of particles in the viewport. Increase the *Display Size* until you can clearly distinguish the axis.

Note

Particles visualized like Point, Circle, Cross and Axis do not have any special options, but can be very useful when you have multiple particle systems at play, if you do not want to confuse particles of one system from another (e.g. in simulations using *Boids* physics).

Color

The Color Menu allows you to display particle's color according to certain particle properties.

None

Particles are black.

Material

Particles are colored according to the material they are given.

Velocity

Color particles according to their speed. The color is a ramp from blue to green to red, Blue being the slowest, and Red being velocities approaching the value of *Max* or above. Increasing *Max* allows for a wider range of particle velocities.

Acceleration

Color particles according to their acceleration.

Amount

Specifies the percentage of all particles to show in the viewport (all particles are still rendered).

Show Emitter

Make instancer visible in viewport.

Children

Reference

Panel:: *Particle System ‣ Children*

Children are *Hair* or *Emitter* particles originating from individual particles. They make it possible to work primarily with a relatively low amount of Parent particles, for whom the physics are calculated. The children are then aligned to their parents. The number and visualization of the children can be changed without a recalculation of the physics.

If you activate children, the parents are no longer rendered. This can be enabled in the Render panel Parent Particles. By default, parent particles are not rendered because the shape of the children can be quite different from that of their parents.

Common Options

Child Type

None
: No children are generated.

Simple
: Children are emitted from the parent position.

Interpolated
: Children are emitted between the *Parent* particles on the faces of a mesh. They interpolate between adjacent parents. This is especially useful for fur, because you can achieve an even distribution. Some of the children can become virtual parents, which are influencing other particles nearby.

Display Amount

The number of children in the 3D Viewport.

Render Amount
The number of children to be rendered.

Length
Length of child paths.

Threshold
Amount of particles left untouched by child path length.

Seed
Offset in the random number table for child particles, to get a different randomized result.

Clumping

Reference

Panel:: *Particle System ‣ Children ‣ Clumping*

Use Clump Curve
Use Curve Widget instead of parameters.

Clump
Clumping amount along child strands. The children may meet at their tip (1.0) or start together at their root (-1.0).

Shape
Form of *Clump*. Either inverse parabolic (0.99) or exponentially (-0.99).

Twist
Todo.

Use Twist Curve
Todo.

Clump Noise

Creates random clumps around the parent hair.

Clump Noise Size
The size of the clumps.

Roughness

Reference

Panel:: *Particle System ‣ Children ‣ Roughness*

Use Roughness Curve
Use Curve Widget instead of parameters.

Uniform, Size
It is based on children location so it varies the paths in a similar way when the children are near.

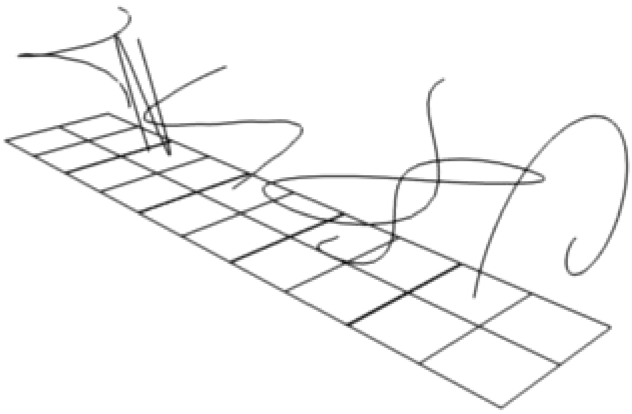
Endpoint, Shape
"Rough End" randomizes path ends (a bit like random negative clumping). Shape may be varied from <1 (parabolic) to 10.0 (hyperbolic).

Random, Size, Threshold
It is based on a random vector so it is not the same for nearby children. The threshold can be specified to apply this to only a part of children. This is useful for creating a few stray children that will not do what others do.

Kink

Reference

Panel:: *Particle System ‣ Children ‣ Kink*



Child particles with Kink.

From left to right: Curl, Radial, Wave, Braid, Spiral.

With *Kink* you can rotate the children around the parent. See Fig. Child particles with Kink. above picture for the different types of *Kink*.

Kink Type

Nothing

Deactivated.

Curl

Children grow in a spiral around the parent hairs.

Radial

Children form around the parent a wave shape that passes through the parent hair.

Wave

Children form a wave, all in the same direction.

Braid

Children braid themselves around the parent hair.

Spiral
Generates a spiral at the end of each hair.

Radius, Resolution
Define the overall size.

Shape
Makes the spiral grow in- or outward.

Note

Alignment Limitations

When hair is pointing straight up (along the chosen spiral axis, default Z), spirals may not show up! This is a limitation of the projection method used. Giving a slight tilt or random orientation to hairs fixes this.

Amplitude
The amplitude of the offset.

Clump
How much clump effects kink amplitude.

Flatness
How flat the hairs are.

Frequency
The frequency of the offset (1/total length). The higher the frequency the more rotations are done.

Shape
Where the rotation starts (offset of rotation).

Simple

Size

A multiplier for children size.

Random Size
Random variation to the size of child particles.

Radius
The radius in which the children are distributed around their parents. This is 3D, so children may be emitted higher or lower than their parents.

Roundness
The roundness of the children around their parents. Either in a sphere (1.0) or in-plane (0.0).

Interpolated

Virtual Parents
Relative amount of virtual parents.

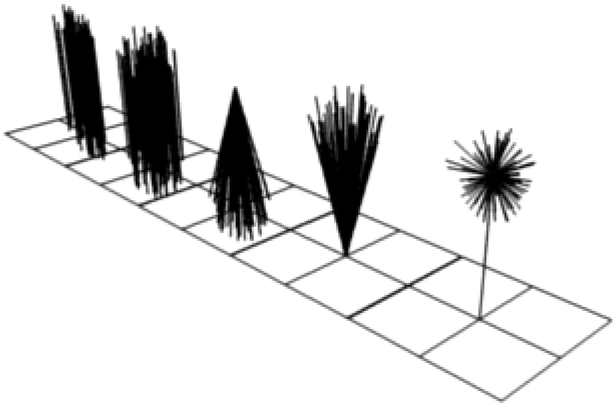
Long Hair
Calculate children that suit long hair well.

Parting

Parting
Creates parting in the children based on parent strands.

Min/Max
The minimum/maximum root to tip angle (tip distance/root distance for long hair).

Example



From left to right: Round: 0.0, Round: 1.0, Clump: 1.0, Clump: -1.0, Shape: -0.99.

Force Fields

Field Weights

Reference

Panel:: *Particle System ‣ Field Weights*

The Field Weight panel allows you to control how much influence each type of external force field, or effector, has on the particle system. Force fields are external forces that give dynamic system's motion. The force fields types are detailed on the Force Field Page.

Effector Group
: Limit effectors to a specified group. Only effectors in this group will have an effect on the current system.

Gravity
: Control how much the Global Gravity has an effect on the system.

All
: Scale all of the effector weights.

Force Fields Settings

Reference

Panel:: *Particle System ‣ Force Fields Settings*

The Force Field Settings panel allows you to make each individual act as a force field, allowing them to affect other dynamic systems, or even, each other.

Self Effect

Causes the particle force fields to have an effect on other particles within the same system.

Effector Amount

Set how many of the particles act as force fields. 0 means all of them are effectors.

You can give particle systems up to two force fields. By default they do not have any. Choose an effector type from the selector to enable them. Settings are described in the Common Settings section.

Vertex Groups

Reference

Panel:: *Particle System ‣ Vertex Groups*

The Vertex groups panel allows you to specify vertex groups to use for several child particle settings. You can also negate the effect of each vertex group with the checkboxes. You can affect the following attributes:

Density
: Defines the density of the particle distribution.

Length
: Defines the length of the hair.

Clump
: Controls the amount of clumping. The weight of 1.0 gives current *Clump* value, weight of 0.0 completely removes effect.

Kink
: Controls the frequency of the children Kink.

Roughness 1
: Adjusts the *Uniform* roughness parameter.

Roughness 2
: Adjusts the *Random* roughness parameter.

Roughness End
: Adjusts the *Endpoint* roughness parameter.

Twist
: Vertex group to control the children's *Twist* effect. Gives control over the direction of the twist, as well as the amount. The weight of 0.5 is

neutral, i.e. there is no twist effect.

Hair

This page is about the end of life hair system. Read about the new hair system on the Hair Nodes page.

- Introduction
    - Growing
    - Styling
    - Animating
    - Rendering
- Emission
- Hair Dynamics
    - Collisions
    - Structure
    - Volume
- Render
- Shape
- Children
- Viewport Display

Introduction

Hair type particle system can be used for strand-like objects, such as hair, fur, grass, quills, etc.

Particle hair systems example. Used for the grass and fur.

Growing

The first step is to create the hair, specifying the amount of hair strands and their lengths.

The complete path of the particles is calculated in advance. So everything a particle does a hair may do also. A hair is as long as the particle path would be for a particle with a lifetime of 100 frames. Instead of rendering every frame of the particle animation point by point there are calculated control points with an interpolation, the segments.

Styling

The next step is to style the hair. You can change the look of base hairs by changing the Physics Settings.

A more advanced way of changing the hair appearance is to use Children. This adds child hairs to the original ones, and has settings for giving them different types of shapes.

You can also interactively style hairs in Particle Edit Mode. In this mode, the particle settings become disabled, and you can comb, trim, lengthen, etc. the hair curves.

Animating

Hair can be made dynamic using the cloth solver. This is covered in the Hair Dynamics page.

Rendering

With Cycles you can render hair with specialized hair BSDFs Hair BSDF or Principled Hair BSDF.

Hair can also be used as a basis for the Particle Instance Modifier, which allows you to have a mesh be deformed along the curves, which is useful for thicker strands, or things like grass, or feathers, which may have a more specific look.

Emission

Reference

Panel:: *Particle System ‣ Emission*

Number

Sets the amount of hair strands. Use as few particles as possible (especially if you plan to use soft body animation later), but still enough to cover the surface and have good control. A few thousand particles is generally enough for a regular haircut. The hair will be made denser later on using Children.

Hair Length

Controls the length of the hair.

See also

Emitter particles Emission panel

Hair Dynamics

Reference

Panel:: *Particle System ‣ Hair Dynamics*

Hair particles can have dynamic properties using physics. To enable hair physics, click the checkbox beside *Hair Dynamics*.

Quality Steps
: Quality of the simulation in steps per frame (higher is better quality but slower).

Pin Goal Strength
: Spring stiffness of the vertex target position.

Warning

If you use motion blur in your animation, you will need to bake one extra frame past the last frame which you will be rendering.

Collisions

Quality
: A general setting for how fine and good a simulation you wish. Higher numbers take more time but ensure less tears and penetrations through the hair.

Distance
: The distance another object must get to the hair for the simulation to repel the hair out of the way. Smaller values might cause errors but provide some speed-up while larger will give unrealistic results if too large and can be slow. It is best to find a good in between value.

Impulse Clamping
: Prevents explosions in tight and complicated collision situations by restricting the amount of movement after a collision.

Collision Collection
: Only objects that are a part of this Collection can collide with the hair. Note that these objects must also have Collision physics enabled.

Structure

Reference

Panel:: *Particle System ‣ Hair Dynamics ‣ Structure*

Vertex Mass
: Value for the mass of the hair.

Stiffness
: Controls the bending stiffness of the hair strands.

  Random
  : Random stiffness of hair.

Damping
: Damping of bending motion.

Volume

Reference

Panel:: *Particle System ‣ Hair Dynamics ‣ Volume*

Some phenomena of real-world hair can be simulated more efficiently using a volumetric model instead of the basic geometric strand model. This means constructing a regular grid such as those used in fluid simulations and interpolating hair properties between the grid cells.

Air Drag
: Controls how thick the air is around the hair causing the hair to flow slower.

Internal Friction
: Amount of friction between individual hairs.

Voxel Grid Cell Size
: Size of the voxel grid cells for interaction effects.

Density Target
: Maximum density of the hair.

Density Strength
: The influence that the *Density Target* has on the simulation.

Render

Reference

Panel:: *Particle System ‣ Render*

Hair can be rendered as a Path, Object, or Group. See Particle Visualization for descriptions and the Hair Shape settings.

See also

Blender Hair Basics, a thorough overview of all of the hair particle settings.

Shape

Reference

Panel:: *Particle System ‣ Hair Shape*

These settings control the shape of hair curves for rendering.

Strand Shape
: A shape parameter that controls the transition in thickness between the root and tip. Negative values make the primitive rounded more towards the top, the value of zero makes the primitive linear, and positive values make the primitive rounded more towards the bottom.

Diameter Root
: Multiplier of the hair width at the root.

Tip
: Multiplier of the hair width at the tip.

Radius Scale
: Multiplier for the *Root* and *Tip* values. This can be used to change the thickness of the hair.

Close Tip
: Sets the thickness at the tip to zero, even when using a nonzero tip multiplier.

Children

Reference

Panel:: *Particle System ‣ Children*

See Children.

Viewport Display

Reference

Panel::	*Particle System ‣ Display*

Rendered
Display hair as curves.

Path
Display just the end points of the hairs.

Steps
The number of segments (control points minus 1) of the hair strand. In between the control points the segments are interpolated. The number of control points is important:

- For the soft body animation, because the control points are animated like vertices, so more control points mean longer calculation times.
- For the interactive editing, because you can only move the control points (but you may recalculate the number of control points in *Particle Edit Mode*).

Hint

Segments

Ten Segments should be sufficient even for very long hair, five Segments are enough for shorter hair, and two or three segments should be enough for short fur.

Texture Influence

Reference

Panel:: *Texture* ▸ *Influence*

Defines the settings of a Particle system spatial with a texture.

General

Time
: Affect the emission time of the particles.

Lifetime
: Affect the life time of the particles.

Density
: Affect the density of the particles.

Size
: Affect the particles size.

Physics

Velocity
: Affect the particles initial velocity.

Damp
: Affect the particles velocity damping.

Gravity
: Affect the particles gravity.

Force Fields

Affect the particles force fields.

Hair

Length
Affect the child hair length.

Clump
Affect the child clumping.

Kink
Affect the child kink.

Rough
Affect the child roughness.

Particle Edit Mode

Using *Particle Edit Mode* you can edit the keyed points (keyframes) and paths of Hair, Particle, Cloth, and Soft Body simulations. (You can also edit and style hair before baking.)

Since working in Particle Edit Mode is pretty easy and very similar to working with vertices in the 3D Viewport, we will show how to set up a particle system and then give a reference of the various functions.

Important

Particle Edit Mode, specifically for hair is deprecated; please use the new Empty Hair object with its associated Sculpt Mode instead.

Important

Editing a cached cloth simulation is not currently working, see: blender/blender#77114 for details.

Usage

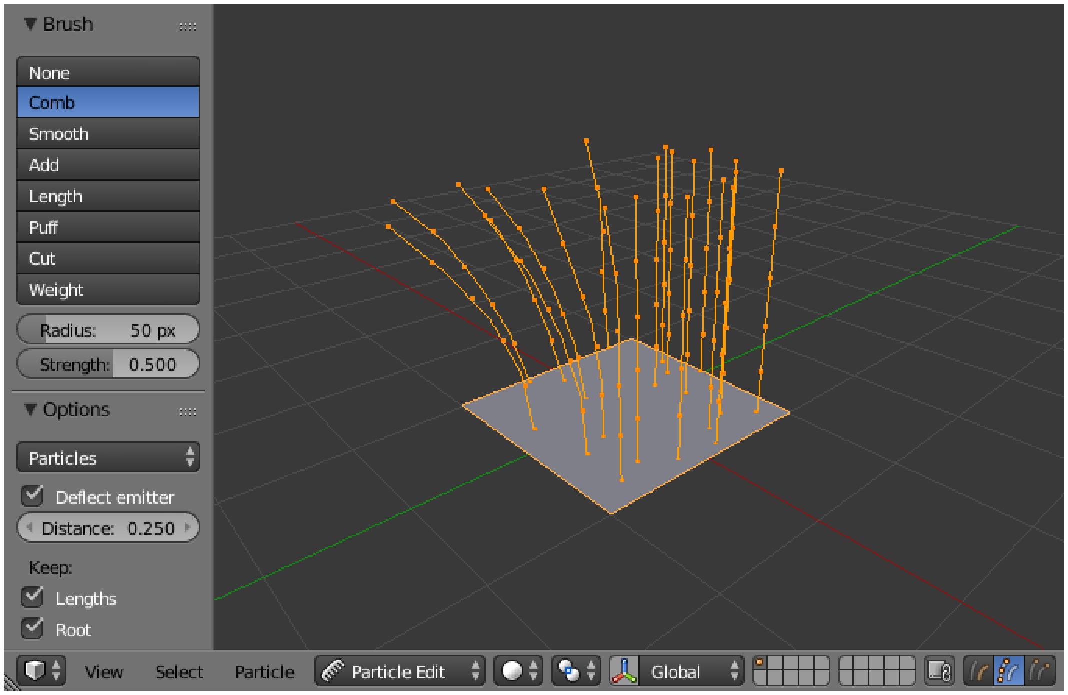
Tip

Only Frames Baked to Memory are Editable!

If you cannot edit the particles, check that you are not baking to a Disk Cache.

Setup for Hair Particles

1. Create a *Hair* particle system.
2. Give it an initial velocity in the *Normal* direction.
3. Create a simulation.
4. Check the *Hair Dynamics* box.


Brush
None
Comb
Smooth
Add
Length
Puff
Cut
Weight
Options
Particles
View
Select
Particle
Particle Edit
Global

Radius: 50 px
Strength: 0.500
☑ Deflect emitter
Distance: 0.250
Keep:
☑ Lengths
☑ Root

Editing hair strands in Particle Edit Mode.

Setup for Particle, Cloth, and Soft Body Simulations

1. Use *Emitter* particles, or a cloth/soft body simulation.
2. Create a simulation by setting up objects and or emitters, set your time range (use a small range if you are just starting out and experimenting), set up the simulation how you want it, using `Alt-A` to preview it.

Bake the Simulation

Once you are happy with the general simulation, bake the simulation from Object Mode. The simulation must be baked to enable editing.

Edit the Simulation

Switch to *Particle Edit* from the *Mode* select menu in the header of the 3D Viewport to edit the particle's paths/Keyframes. You may need to press `T` from within the 3D Viewport to see the *Particle Edit* toolbox. Move to the frame you want to edit and use the various tools to edit your simulation.

Selecting

Tip

Switch to the *Point select mode* (see below) in the header of the 3D Viewport to be able to see and select the keypoints.

- Select single: `LMB`.
- Add to/remove from selection: `Shift-LMB`.
- All: `A`.
- None: `Alt-A`.
- Invert: `Ctrl-I`.
- Box select: `B`.
- Circle Select: `C`.
- Lasso Select: `Ctrl-Alt-LMB`.
- Select Linked: Move the mouse over a path and press `L` to add all its points to the selection.
- Unselect Linked: Move the mouse over a path and press `Shift-L` to remove all its points from the selection.
- Root/Tips: *Select ‣ Roots / Tips*.

Select Random

Randomly selects particles.

Percent
  Percent of particles to randomly select.

Random Seed
  Seed value to use for the selection.

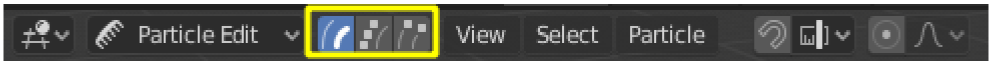
Action

Select random can be either used to select or deselect particles.

Type

Selects either hair or points. Here these terms can be confusing because hair/point does not refer to the particle type but the path/points of the hair/particle.

Select Modes



Select Modes.

Path::	No keypoints are visible, you can select/deselect only all particles.
Point::	You see all of the keypoints.
Tip::	You can see and edit (including the brushes) only the tip of the particles, i.e. the last keypoint.

Tools

Reference

Mode::	Particle Edit Mode
Tool::	*Toolbar*

Comb

Moves the keypoints (similar to the Proportional Editing tool).

Deflect Emitter

Hair particles only – Do not move keypoints through the emitting mesh.

Distance

The distance to keep from the Emitter.

Smooth

Parallels visually adjacent segments.

Add

Adds new particles.

Count
: The number of new particles per step.

Interpolate
: Interpolate the shape of new hairs from existing ones.

Steps
: Amount of brush steps.

Keys
: How many keys to make new particles with.

Length

Scales the segments, so it makes the hair longer with *Grow* or shorter with *Shrink*.

Grow/Shrink
: Sets the brush to add the effect or reverse it.

Puff

Rotates the hair around its first keypoint (root). So it makes the hair stand up with *Add* or lay down with *Sub*.

Puff Volume
: Apply puff to unselected end points, (Helps to maintain the hair volume when puffing the root.)

Cut

Scales the segments until the last keypoint reaches the brush.

Weight

This is especially useful for soft body animations, because the weight defines the soft body *Goal*. A keypoint with a weight of 1 will not move at all, a keypoint with a weight of 0 subjects fully to soft body animation. This value is scaled by the Strength *Min* to *Max* range of soft body goals…

Common Options

Below the brush types, their settings appear:

Radius `F`
: Set the radius of the brush.

Strength `Shift-F`
: Set the strength of the brush effect (not for Add brush).

Options

Reference

Mode::	Particle Edit Mode
Panel::	*Tool Settings* ‣ *Options*

Auto-Velocity *Emitter*
: Recalculate velocities of particles according to their edited paths. Otherwise, the original velocities values remains unchanged regardless of the actual distance that the particles moves.

Mirror X
: Enable mirror editing across the local X axis.

Preserve

Strand Length

Keep the length of the segments between the keypoints when combing or smoothing the hair. This is done by moving all the other keypoints.

Root Positions

Keep first key unmodified, so you cannot transplant hair.

Cut Particles to Shape

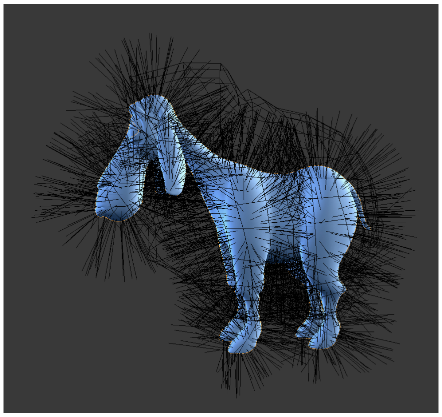
Shape Object

A mesh object which boundary is used by the *Shape Cut* tool.

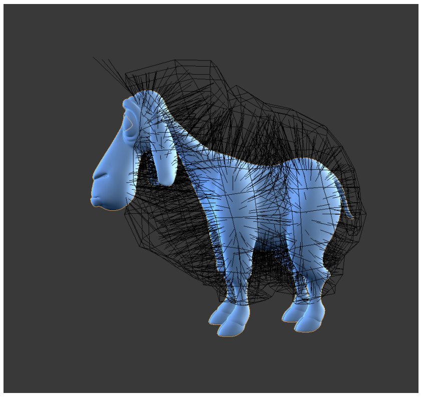
Cut

This grooming tool trims hairs to a shape defined by the *Shape Object*. This is a quicker way of avoiding protruding hair sections from lengthening than using the Cutting tool. It works especially well for characters with extensive fur, where working in a single plane with the Cutting tool becomes tedious.

Shape Cut example.



Before.



After.

Viewport Display

Path Steps
  The number of steps used to draw the path; improves the smoothness of the particle path.

Children *Hair*
  Displays the children of the particles too. This allows to fine-tune the particles and see their effects on the result, but it may slow down your system if you have many children.

Particles *Emitter*
  Displays the actual particles on top of the paths.

Fade Time
  Fade out paths and keys further away from current time.

  Frames
    How many frames to fade.

Editing

Moving Keypoints or Particles

- To move selected keypoints press `G`, or use one of the various other methods to move vertices.
- To move a particle root you have to turn off Keep *Root* in the Toolbar.
- You can do many of the things like with vertices, including scaling, rotating and removing (complete particles or single keys).
- You may not duplicate or extrude keys or particles, but you can subdivide particles which adds new keypoints *Particle ‣ Subdivide*.
- Alternatively you can re-key a particle *Particle ‣ Rekey*.

How smoothly the hair and particle paths are displayed depends on the *Path Steps* setting in the Toolbar. Low settings produce blocky interpolation between points, while high settings produce a smooth curve.

Mirror

Reference

Mode:: Particle Edit Mode
Menu:: *Particle ▸ Mirror*

If you want to create an X axis symmetrical haircut you have to do following steps:

1. Select all particles with `A`.
2. Mirror the particles with *Particle ▸ Mirror*.
3. Turn on *X Mirror* in *Sidebar Region ▸ Tool ▸ Options*.

It may happen that after mirroring two particles occupy nearly the same place. Since this would be a waste of memory and render time, you can use *Merge by Distance* from the *Particle* menu.

Unify Length

Reference

Mode:: Particle Edit Mode
Menu:: *Particle ▸ Unify Length*

This tool is used to make all selected hair uniform length by finding the average length.

Show/Hide

Reference

Mode:: Particle Edit Mode
Menu:: *Particle ▸ Show/Hide*

Hiding and unhiding of particles works similar as with vertices in the 3D Viewport. Select one or more keypoints of the particle you want to hide and press `H`. The particle in fact does not vanish, only the key points.

Hidden particles (i.e. particles whose keypoints are hidden) do not react on the various brushes. But:

If you use *Mirror Editing* even particles with hidden keypoints may be moved, if their mirrored counterpart is moved.

To unhide all hidden particles press `Alt-H`.

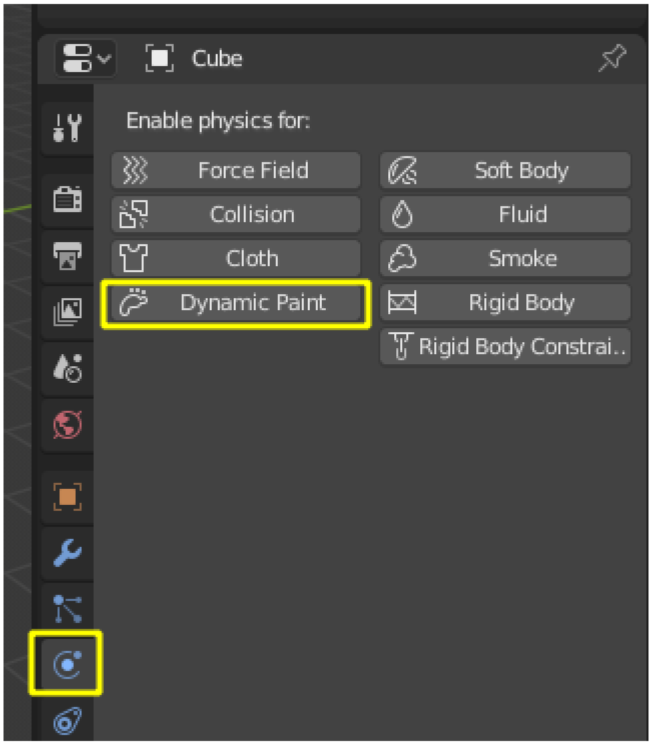
Dynamic Paint

- Introduction
    - Activating the Modifier
    - Types
- Brush
    - Source
    - Velocity
    - Waves
- Canvas
    - Settings
    - Surface
    - Cache
    - Effects
    - Initial Color
    - Output

Introduction

Dynamic paint is a modifier and physics system that can turn objects into paint canvases and brushes, creating Color Attributes, image sequences, or displacement. This makes many effects possible like, for example footsteps in the snow, raindrops that make the ground wet, paint that sticks to walls, or objects that gradually freeze.

Activating the Modifier


Cube
Enable physics for:
Force Field
Soft Body
Collision
Fluid
Cloth
Smoke
Dynamic Paint
Rigid Body
Rigid Body Constrai..

How to activate the Dynamic Paint.

Dynamic Paint can be activated from the “Physics” tab of the “Properties” editor.

Types

Modifier itself has two different types Canvas and Brush.

Note

You can also enable brush and canvas simultaneously. In that case same object's "brush" does not influence its "canvas", but can still interact with other objects in the scene.

See also

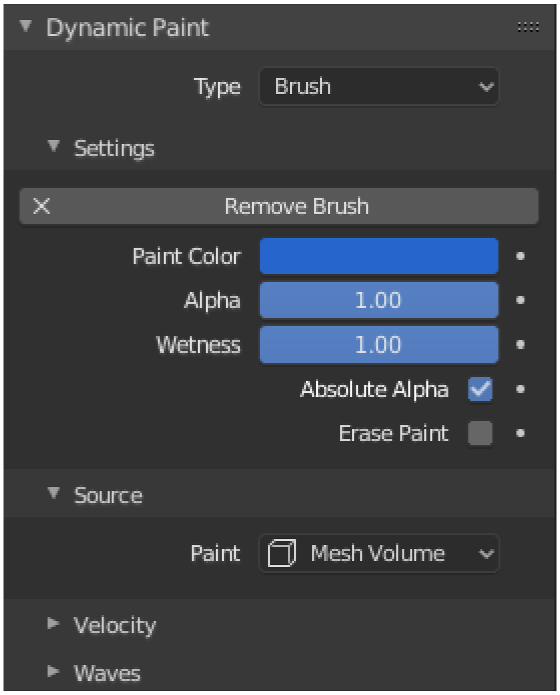
- A step-by step introduction.
- A detailed guide that covers every setting with images and examples (currently not up-to-date).

Brush

Reference

Panel:: *Physics* ▸ *Dynamic Paint*
Type:: Brush

The Brush type makes object apply paint on the canvas.



Dynamic Paint
Settings
Remove Brush
Source
Velocity
Waves

Type: Brush
Paint Color
Alpha: 1.00
Wetness: 1.00
☑ Absolute Alpha
☐ Erase Paint
Paint: Mesh Volume

Brush main panel.

From the first brush panel you can define how brush affects canvas color surfaces.

Paint Color
: Color of the paint.

Alpha
: Defines brush alpha or visibility. Final wetness is also affected by alpha.

Wetness

Defines how "wet" new paint is. Wetness is visible on "Paint" surface "wetmap". Speed of "Drip" and "Spread" effects also depends on how wet the paint is.

Absolute Alpha

This setting limits brush alpha influence. Without it, brush is "added" on surface over and over again each frame, increasing alpha and therefore influence of brush on canvas. In many cases however, it is preferred to not increase brush alpha if it already is on brushes level.

Erase Paint

Makes brush dissolve existing paint instead of adding it.

Source

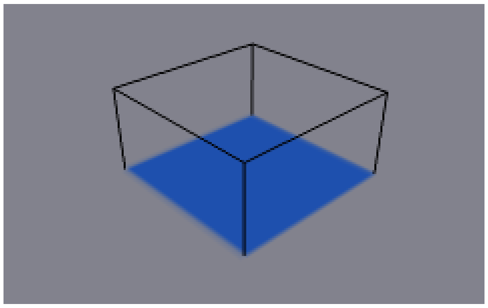
Reference

Type::	Brush
Panel::	*Physics* ‣ *Dynamic Paint* ‣ *Source*

Paint source setting lets you define how brush influence/intersection is defined.

Mesh Volume

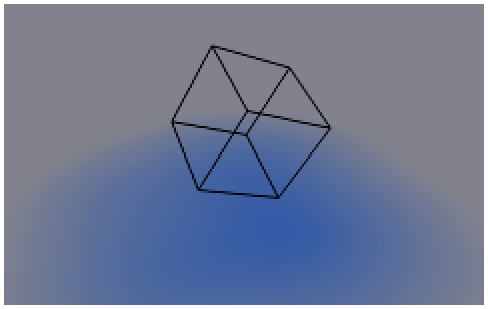
The Brush affects all surface point inside the mesh volume.



Source: Mesh Volume.

Proximity

Only uses defined distance to the closest point on brush mesh surface. Note that inside of the volume is not necessarily affected because it is not close to the surface.



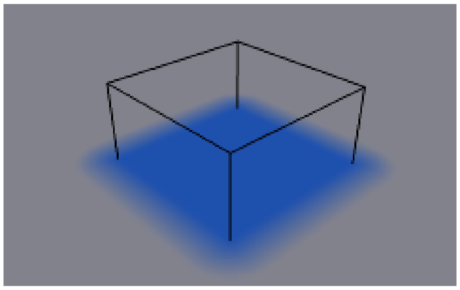
Source: Proximity. Brush affects all canvas pixels around it.

Mesh Volume + Proximity

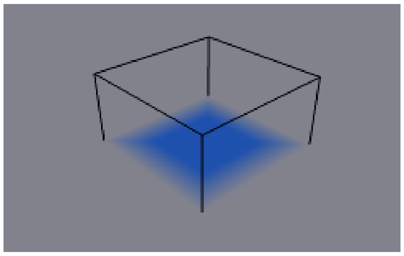
Same as volume type, but also has influence over defined distance.

Inner Proximity
Applies proximity inside the mesh volume.

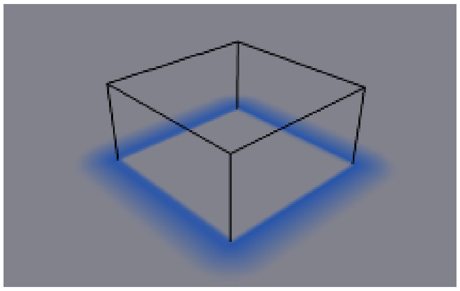
Negate Volume
Negates brush alpha within mesh volume.



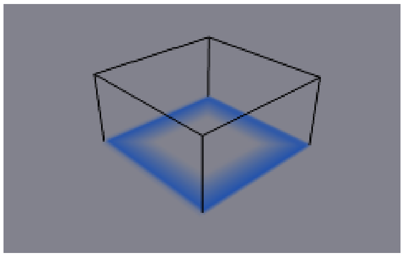
The Volume + Proximity brush with no additional settings.



Inner Proximity. Proximity falloff is now visible inside the volume.



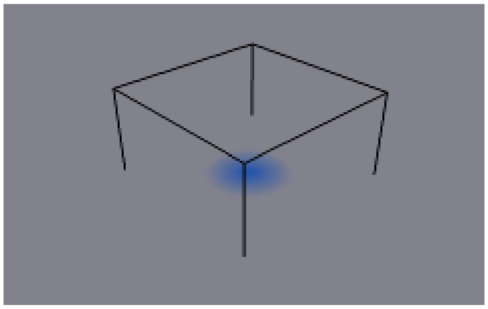
Negate Volume. Inner side of the volume has become completely transparent.



Inner Proximity and Negate Volume enabled together.

Object Center

Instead of calculating proximity to the brush object mesh, which can be quite slow in some cases, only distance to only center is calculated. This is much faster and often good enough.



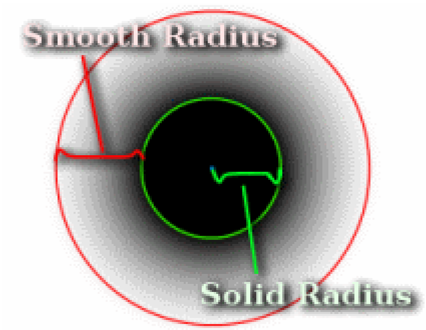
Source: Object Center.

Particle System

Brush influence is defined by particles from a selected particle system.

Effect Solid Radius
: Defines the distance, inside which paint is solid color.

Use Particle Radius

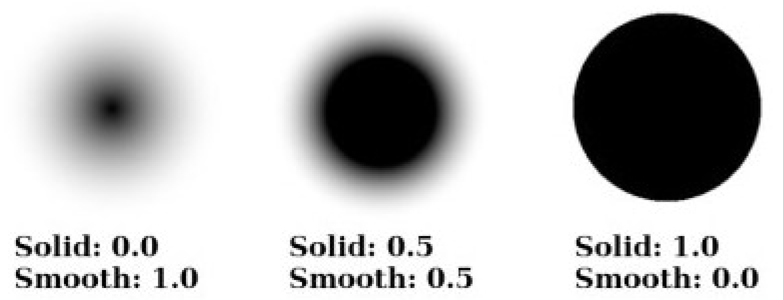

Smooth Radius
Solid Radius

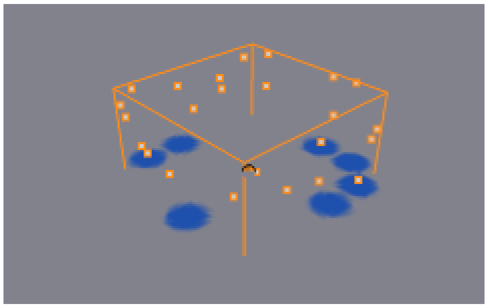
Uses the settings in the particle panel to determine solid radius size. Solid Radius size disabled while Particle Radius enabled.

Smooth Radius

An additional radius outside Solid Radius to add a smooth falloff.

If you set “Smooth Radius” to zero, particle will be painted as a solid sphere. If you set “Solid Radius” to zero, it gets painted as a smooth halo.


Solid: 0.0
Smooth: 1.0
Solid: 0.5
Smooth: 0.5
Solid: 1.0
Smooth: 0.0



Source: Particle System.

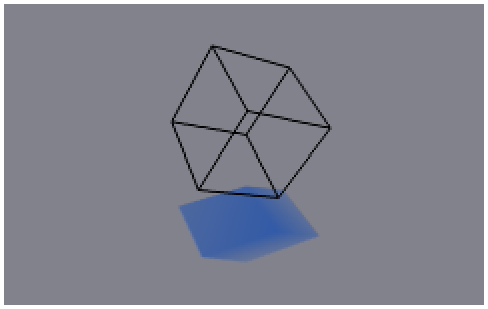
Common Options

Paint Distance

The maximum distance to mesh surface to affect paint.

Project

Projects brush to the canvas from a defined direction. Basically this can be considered as “direction aligned” proximity.



The Project option enabled. See how brush only affects canvas in normal direction.

Falloff

Sharp::	Paints solid paint within the defined distance.
Smooth::	Makes paint to linearly fade out until becoming completely invisible when it reaches the maximum distance.
Color Ramp::	Allows you to manually make a custom falloff behavior.

Velocity

Reference

Type::	Brush
Panel::	*Physics ‣ Dynamic Paint ‣ Velocity*

This panel shows brush options that are based on object velocity.

On top you have a color ramp and several related settings. Basically the color ramp represents brush velocity values: left side being zero velocity and right side being the “Max velocity”. Speed is measured in “units per frame”.

Checkboxes above can be used to define color ramp influence.

Multiply Alpha

Uses color ramp’s alpha value depending on current velocity and multiplies brush alpha with it.

Replace Color

Replaces the brush color with the values from the Color Ramp Widget.

Multiply Depth

Multiplies brushes “depth intersection” effect. Basically you can adjust displace and wave strength depending on brush speed.

Do Smudge

Enabling Smudge makes the brush “smudge” (or “smear”) existing colors on the surface as it moves. The strength of this effect can be defined from the *Smudge Strength* property.

Even when smudge is enabled brush still does its normal paint effect. If you want a purely smudging brush use zero alpha. It is also possible to have *Erase* option enabled together with smudge.

Waves

Reference

Type::	Brush
Panel::	*Physics ‣ Dynamic Paint ‣ Waves*

This panel is used to adjust brush influence to “Wave” surfaces.

Wave Type

Select what effect the brush creates in the wave simulation.

Depth Change::	The brush create waves when the intersection depth with the surface is *changed* on that point. If the brush is not moved, it will have no effect.

	Using a negative “Factor” with this type can create a nice looking “wake” for moving objects like ships.
Obstacle::	Constantly affects surface whenever intersecting. Waves are also reflected off this brush type. However, due the nature of wave simulation algorithm this type creates an unnatural “dent” in the surface if the brush is not moved.
Force::	Directly affects the velocity of wave motion. Therefore the effect is not one-to-one with brush intersection depth, yet the force strength depends on it.
Reflect Only::	This type has no visible effect on the surface alone but reflects waves that are already on the surface.

Factor

Adjusts how strongly brush “depth” affects the simulation. You can also use negative values to make brush pull water up instead of down.

Clamp Waves

In some cases the brush goes very deep inside the surface messing whole simulation up. You can use this setting to “limit” influence to only certain depth.

Canvas

Reference

Panel::	*Physics* ▸ *Dynamic Paint*
Type::	Canvas

The Canvas type makes object receive paint from Dynamic Paint brushes.

Settings

Paint Surface

A list of Dynamic Paint surfaces. These surfaces are basically layers of paint, that work independently from each other. You can define individual settings for them and bake them separately.

Is Active

The checkbox toggles whether surface is active at all. If not selected no calculations are done.

Below you can set surface type and adjust quality and timing settings.

Format

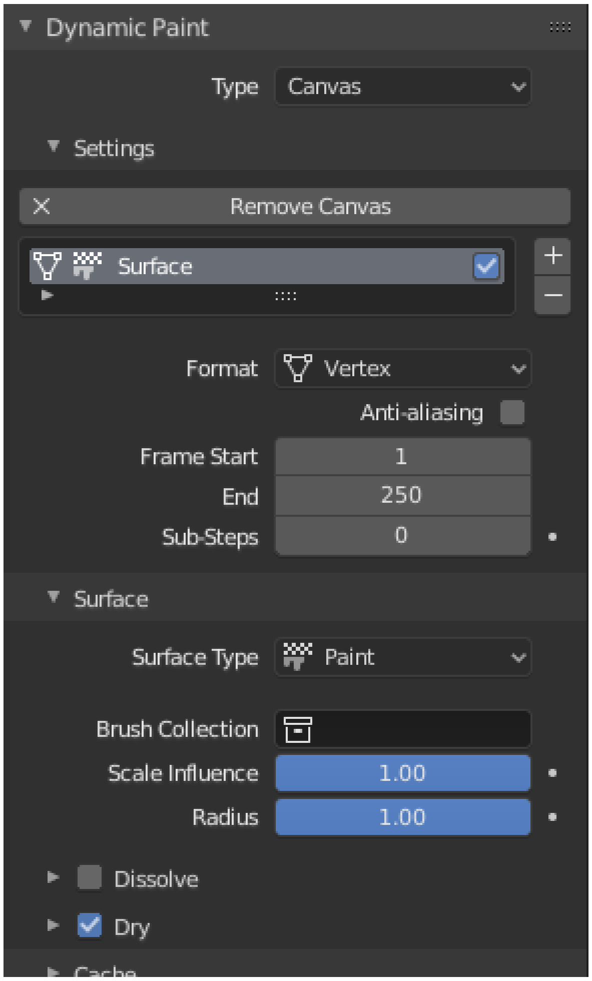
Each surface has a certain format and type. Format determines how data is stored and outputted.

Dynamic Paint operates directly on mesh vertex data. Results are stored by point cache and can be displayed in viewports. However, using vertex level also requires a highly subdivided mesh to work.

Dynamic Paint generates UV wrapped image files of defined resolution as output.
Resolution

Vertex::
Image Sequences::

You can adjust the output image dimensions for the *Image Sequences* surface type. For example using 256 will lead to 256×256 image output. Doubling the resolution will likely quadruple the baking time and vice versa.


Dynamic Paint
Settings
Remove Canvas
Surface
Surface
Brush Collection
Cache
Type: Canvas
Surface ☑
Format: Vertex
Anti-aliasing ☐
Frame Start: 1
End: 250
Sub-Steps: 0
Surface Type: Paint
Scale Influence: 1.00
Radius: 1.00
☐ Dissolve
☑ Dry

Canvas main panel.

Anti-Aliasing

[Anti-Aliasing](#) to smooth paint edges using a 5× multisampling method.

Frame Start, End

Defines surface processing start and end frame.

Sub-Steps

Extra samples between frames. They are usually required when there is a very fast brush.

Surface

Reference

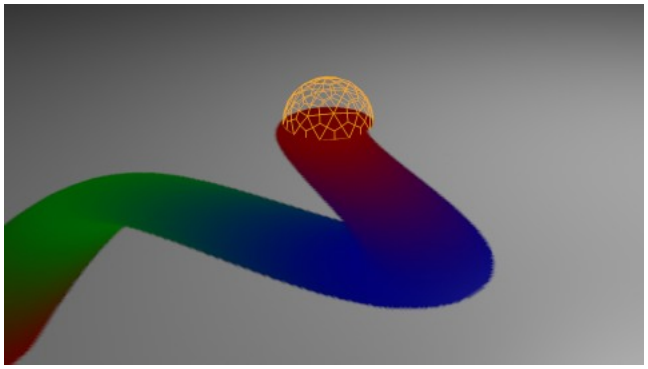
Type:: Canvas
Panel:: *Physics ‣ Dynamic Paint ‣ Surface*

From *Surface* panel you can adjust surface type and related settings.

Surface Type

Each surface has a “type” that defines what surface is used for.

Paint



Paint Surface.

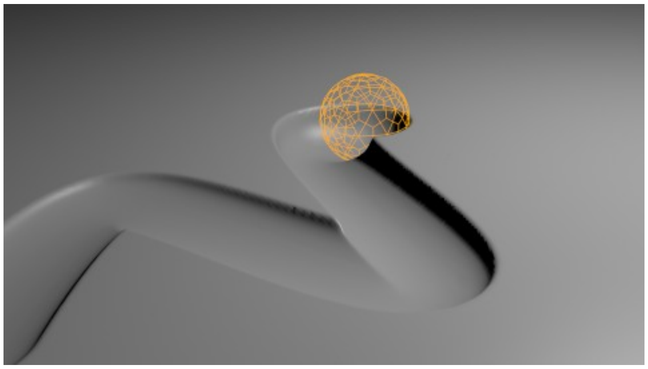
Paint is the basic surface type that outputs color and wetness values. In case of vertex surfaces, results are outputted as Color Attributes.

A wetmap is a black-and-white output that visualizes paint wetness. White being maximum wetness, black being completely dry. It is usually used as mask for rendering. Some “paint effects” affect wet paint only.

Dry
Completely disable drying is useful for indefinitely spreading paint.

Color Dry
It can be used to define wetness level when paint colors start to shift to surface “background”. Lower values can be useful to prevent spreading paint from becoming transparent as it dries, while higher values give better results in general.

Displace



Displace Surface.

This type of surface outputs intersection depth from brush objects.

Incremental
A new displace is added cumulatively on top of an existing displace.

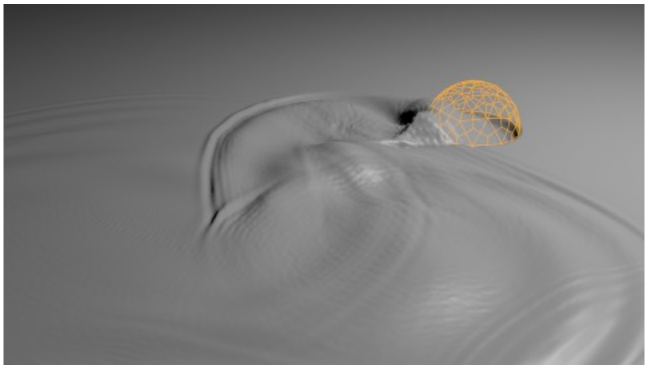
Max Displace
The maximum level of intersection depth, larger values will be clamped to this value.

Displace Factor
The multiplier for the intersection depth. You can use it to adjust final displace strength or use negative values to paint bumps.

Tip

If the displace output seems too rough it usually helps to add a Smooth Modifier after Dynamic Paint in the modifier stack.

Waves



Waves Surface.

This surface type produces simulated wave motion. Like displace, wave surface also uses brush intersection depth to define brush strength.

Open Borders

Allows waves to pass through mesh “edges” instead of reflecting from them.

Timescale

Directly adjusts simulation speed without affecting simulation outcome. Lower values make simulation go slower and otherwise.

Speed

Affects how fast waves travel on the surface. This setting is also corresponds to the size of the simulation. Half the speed equals surface double as large.

Damping

Reduces the wave strength over time. Basically adjusts how fast wave disappears.

Spring

Adjusts the force that pulls water back to “zero level”.

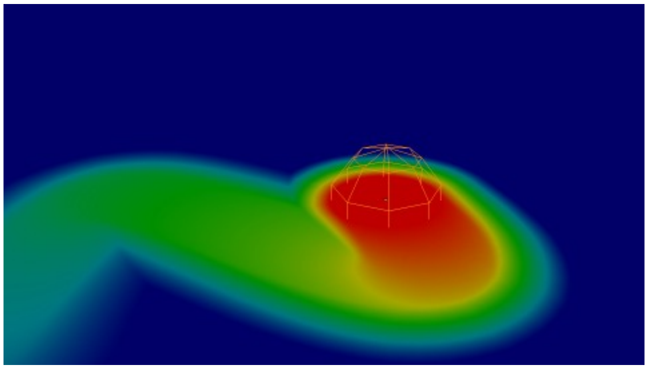
Smoothness

Limits the maximum steepness of the wave slope between simulation points. This greatly helps getting rid of the “noise” that occurs when using objects with sharp edges (like cubes) as a brush. The default value should be enough to only get rid of the sharpest spikes, in order to get even smoother waves use higher values at the expense of reduced detail.

Tip

In some cases the wave motion gets very unstable around brush. It usually helps to reduce wave speed, brush “wave factor” or even the resolution of mesh/surface.

Weight



Weight Surface.

This is a special surface type only available for vertex format. It outputs vertex weight groups that can be used by other Blender modifiers and tools.

Tip

It is usually preferred to use “proximity” based brushes for weight surfaces to allow smooth falloff between weight values.

Common Options

For each surface type there are special settings to adjust. Most types have the settings *Dissolve* and *Brush*:

Dissolve
  Used to make the surface smoothly return to its original state during a defined *Time* period.

Brush Collection
  Used to define a specific collection to pick brush objects from.

Influence Scale, Radius Scale
  For tweaking brush settings individually for each surface.

Cache

Reference

Type::	Canvas
Panel::	*Physics ‣ Dynamic Paint ‣ Cache*

This panel is currently only visible for *Vertex* format surfaces. You can use it to adjust and bake point cache.

Effects

Reference

Type::	Canvas
Panel::	*Physics ‣ Dynamic Paint ‣ Effects*

This is a special feature for “Paint” type surface. It generates animated movement on canvas surface.

Effects
  Spread

Paint slowly spreads to surrounding points eventually filling all connected areas.

Drip
: Paint moves in specific direction specified by Blender force fields, gravity and velocity with user-defined influences.

Shrink
: Painted area slowly shrinks until disappears completely.

For spread and drip effects, only “wet paint” is affected, so as the paint dries, movement becomes slower until it stops.

Initial Color

Reference

Type::	Canvas
Panel::	*Physics ‣ Dynamic Paint ‣ Initial Color*

Allows you to define the initial color of the canvas. (Todo 2.62)

- None
- Color
- UV Texture
- Vertex Color

Output

Reference

Type::	Canvas
Panel::	*Physics ‣ Dynamic Paint ‣ Output*

From Output panel you can adjust how surface outputs its results.

Vertex

For *Vertex* format surfaces, you can select a mesh data layer (color/weight depending on surface type) to generate results to. You can use the “+”/”-” icons to add/remove a data layers of given name. If layer with given name is not found, it is shown as red.

Image Sequence

For *Image Sequence* surfaces, you can define used UV maps and output file saving directory, filenames and image format.

Forces

- Gravity
- Force Fields
    - Introduction
    - Types

Gravity

Reference

Panel:: *Scene* ▸ *Gravity*

Gravity is a global setting that is applied to all physics systems in a scene. It can be found in the scene tab. This value is generally fine left at its default, -9.810 on the Z axis, which is the force of gravity in the real world. Changing this value would simulate a lower or higher force of gravity. Gravity denoted g, measurement $m \times s^{-2}$.

Gravity is applied in the same way to all physics systems.

Gravity is practically the same around the entirety of planet *Earth*. For rendering scenes on The Moon, use -1.622 $m \times s^{-2}$ on the Z axis. Another popular gravity value might be for Mars which has a gravitation acceleration of -3.69 $m \times s^{-2}$ on the Z Axis.

Note

The gravity value per physics system can be scaled down in the *Field Weights* tab.

Force Fields

- Introduction
    - Creating a Force Field
    - Common Field Settings

Types

- Boid
- Charge
- Curve Guide
- Drag
- Fluid Flow
- Force
- Harmonic
- Lennard-Jones
- Magnetic
- Texture
- Turbulence
- Vortex
- Wind

Introduction

Force fields offer a way to influence a simulation, in example to add extra movement. Particles, Soft Bodies, Rigid Bodies, and Cloth objects can all be affected by forces fields. Force fields automatically affect everything. To remove a simulation or particle system from their influence, simply turn down the influence of that type of force field in its Field Weights panel.

- All types of objects and particles can generate fields, but only curve object can bear a Curve Guide field.
- Force fields can also be generated from particles. See Particle Physics.
- The objects need to share at least one common layer to have an effect.

You may limit the effect on particles to a group of objects (see the Particle Physics page).

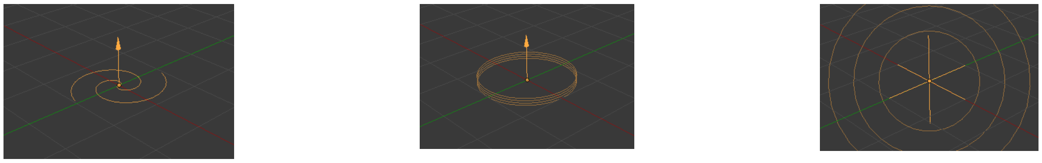
Creating a Force Field

Reference

Mode:: Object Mode
Menu:: *Add ▸ Force Field*
Panel:: *Physics ▸ Force Field*

To create a single force field, you can select *Add ▸ Force Field* and select the desired force field. This method creates an empty with the force field attached.

Examples of an empty with the force field attached.



Vortex force field.	Wind force field.	Force force field.

To create a field from an existing object you have to select the object and change to the *Physics* tab. Select the field type in the *Fields* menu.

Note

After changing the fields *Fields* panel or deflection *Collision* panel settings, you have to recalculate the particle, soft body or cloth system by *Free Cache*, this is not done automatically.

Particles react to all kinds of force fields, soft bodies only to *Force*, *Wind*, *Vortex* (they react on *Harmonic* fields but not in a useful way).

Common Field Settings

Most fields have the same settings, even though they act very differently. Settings unique to a field type are described below. Curve Guide and Texture fields have very different options.

Shape

Sets the direction which is used to calculate the effector force. For force fields from an empty object only *Point*, *Line* and *Plane* shapes are available, as for a field from a 3D object there are additional *Surface* and *Every Point* options, and *Curve* for a field from a curve.

Point::	Point with omni-directional influence. Uses the object origin as the effector point.
Line::	The force only acts in the local XY plane, using the Z axis line as the effector.
Plane::	The force only acts in the local Z direction, using the XY axis plane as the effector.
Surface::	The force field acts on a 3D object's surface. In this case, the Z axis is the surface normal.

Every Point:: Uses every vertex in the mesh object as an effector point.

Strength

The strength of the field effect. This can be positive or negative to change the direction that the force operates in. A force field's strength is scaled with the force object's scale, allowing you to scale up and down the scene, keeping the same effects.

Flow

If nonzero, this adds a drag force proportional and opposite to the point velocity.

This effectively re-interprets the force field so that the *Strength* to *Flow* ratio at a certain point defines the velocity of an "air flow" field, and objects are encouraged to follow the flow by the resistance caused by the *Flow* drag force.

Affect

Location

Influence the location of particles and other physics entities.

Rotation

Influence the rotation of particles with Dynamic Rotation. The option is not relevant for other types of physics systems.

Disabling both options completely deactivates the force field.

Noise Amount

Adds noise to the strength of the force.

Seed

Changes the seed of the random noise.

Absorption

Force gets absorbed by collision objects.

Wind Factor

Specifies how much the force is reduced when acting parallel to a surface, e.g. cloth. If set to 1, only the normal component of the force is taken into account.

Falloff

Here you can specify the shape of the force field (if the falloff *Power* is greater than 0).

Shape

Cone::	The falloff results in a cone-shaped force field. Additional options are the same as those of *Tube* options.
Sphere::	The falloff is uniform in all directions, as in a sphere.
Tube::	The falloff results in a tube-shaped force field. The field's *Radial Power* can be adjusted, as well as the *Minimum* and *Maximum* distances of the field.

Z Direction

The direction the force affects on the Z axis.

+Z::	The force only has an affect on the positive Z axis.
-Z::	The force only has an affect on the negative Z axis.
Both Z::	The force has an affect on the positive and negative Z axis.

Power

How the power of the force field changes with the distance from the force field. If r is the distance from the origin of the object, the force changes with $1/(r - min + 1)^{power}$. A falloff of 2 changes the force field with $1/(r - min + 1)^2$, which is similar to the falloff of gravitational pull.

Min Distance

The distance from the object's origin, up to where the force field is effective with full strength. If you have a falloff of 0, this parameter will have no effect, because the field is effective with full strength up to *Max Distance* (or infinite). Shown by an additional circle around the object.

Max Distance

Specifies the maximum radius in which the force field affects other objects (shown by an additional circle around the object).

Boid

Reference

Panel:: *Physics* ▸ *Force Fields*
Type:: Boid

The *Boid* force fields do not affect physics, and are used together with the Boids Particles to define boid predators and goals for the *Boid Brain* rules.

Charge

Reference

Panel:: *Physics* ▸ *Force Fields*
Type:: Charge

A *Charge* force field is similar to a *Force* field except it changes it’s behavior (attract/repulse) based on the effected particles charge field (negative/positive), like real particles with a charge. Which means that this field has only an effect on particles that have also a *Charge* field (else, they have no “charge”, and hence are unaffected)!

Example

Curve Guide

Reference

Panel:: *Physics* ▸ *Force Fields*
Type:: Curve Guide

The *Curve Guide* is used to force particles to follow a certain path defined by a Curve Object. A typical scenario would be to move a red blood cell inside a vein, or to animate the particle flow in a motor. You can also use *Curve Guide* to shape certain hair strands.

Note

You can also use the Particle Edit Mode to define a path.

Since you can animate curves as a soft body or any other usual way, you may build very complex animations while keeping great control and keeping the simulation time to a minimum.

To make particles point in the direction of the curve, you need to set their *Orientation Axis* to *Velocity / Hair*, enable *Dynamic*, and set their *Angular Velocity Axis* to *Velocity*, all in the the Rotation settings of the particle system. The Follow Path Constraint and the curve's legacy Follow option won't work for this.

A *Curve Guide* force affects all particles on the same layer, independently from their distance to the curve. If you have several guides in a layer, their fields add up to each other (the way you may have learned it in your physics course). But you can limit their influence radius by changing the *Minimum Distance* (see below).

A particle follows a *Curve Guide* during its lifetime, the velocity depends on its lifetime and the length of the path.

Note

The Curve Guide does not affect soft bodies.

Options

Free
: Fraction of particle life time, that is not used for the curve.

Falloff Power
: This setting governs the strength of the guide between *Min Distance* and *Max Distance*. A falloff of 1 means a linear progression.

Additive
: If you use *Additive*, the speed of the particles is also evaluated depending on the falloff.

Weights
: Use Curve weights to influence the particle influence along the curve.

Clumping Amount
: The particles come together at the end of the curve (1) or they drift apart (-1).

Shape
: Defines the form in which the particles come together. +0.99: the particles meet at the end of the curve. 0: linear progression along the curve. -0.99: the particles meet at the beginning of the curve.

Min Distance
: The distance from the curve, up to where the force field is effective with full strength. If you have a falloff of 0, this parameter will have no effect, because the field is effective with full strength up to *Max Distance* (or the infinity). *Min Distance* is shown with a circle at the endpoints of the curve in the 3D Viewport.

Max Distance

The maximum influence radius. Shown by an additional circle around the curve object.

Kink

Warning

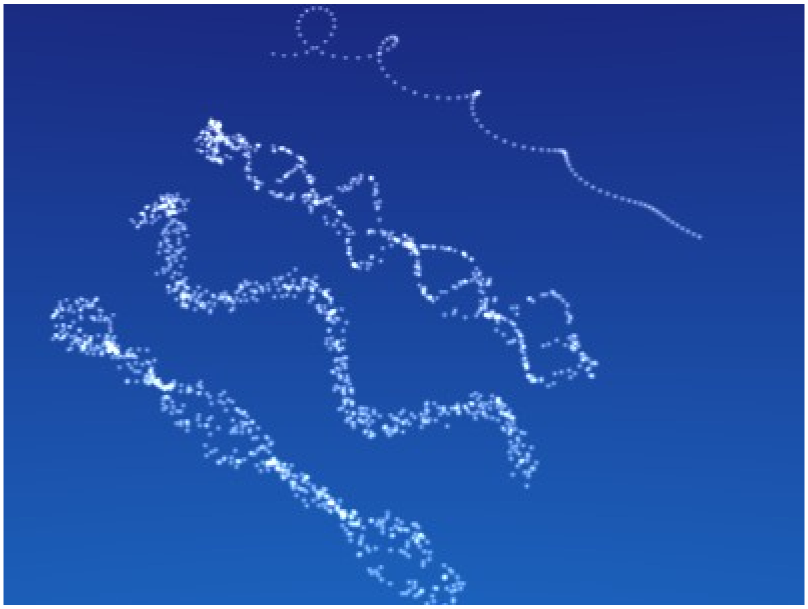
This feature is broken in the current version, see Bug Report #46776.

Type

Changes the shape that the particles can take.

None::	Todo.
Braid::	Braid.
Curl::	The radius of the influence depends on the distance of the curve to the emitter.
Radial::	A three-dimensional, standing wave.
Roll::	A one-dimensional, standing wave.
Rotation::	Todo.
Wave::	A two-dimensional, standing wave.

It is not so easy to describe the resulting shapes, so have a look at the example below.



Kink options of a curve guide. From left to right: Radial, Wave, Braid, Roll. Animation.

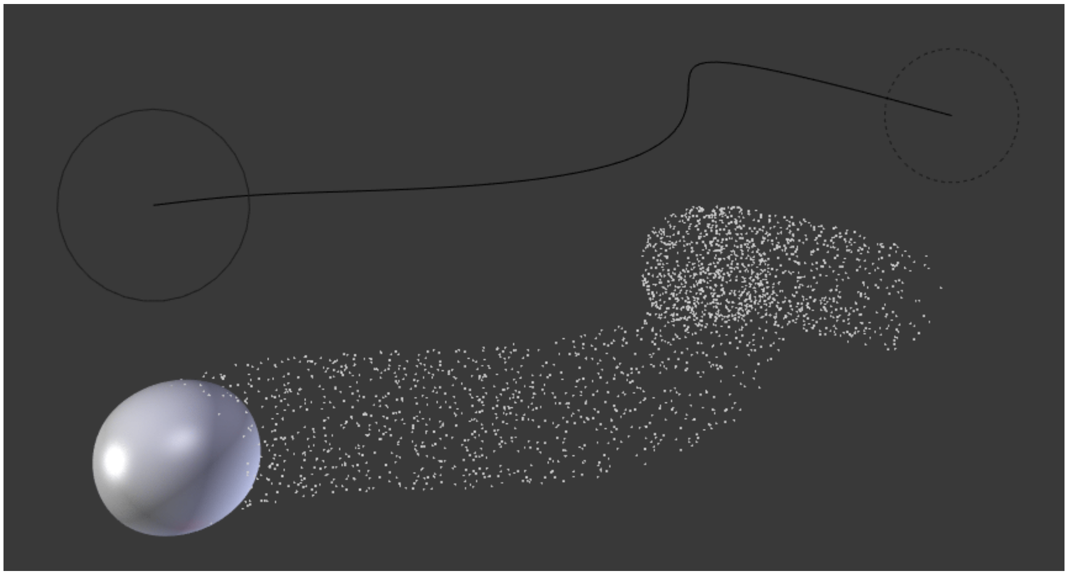
Axis
    Which axis to use for the offset.

Frequency
    The frequency of the offset.

Shape
    Adjust the offset to the beginning/end.

Amplitude
    The Amplitude of the offset.

Examples



Curve Guide force field.

Drag

Reference

Panel::	*Physics* ▸ *Force Fields*
Type::	Drag

A *Drag* force field resists particle motion by slowing it down.

Options

Linear
: Drag component proportional to velocity.

Quadratic
: Drag component proportional to the square of the velocity.

Fluid Flow

Reference

Panel::	*Physics ▸ Force Fields*
Type::	Fluid Flow

The *Fluid Flow* force field creates a force based on a Fluid simulation air flow. It applies the smoke simulation air flow velocity as a force to other simulations that use force fields. To use it you need to add a *Fluid Flow* force field and select a domain object for it. For example fire sparkles can realistically flow along the air turbulence near the simulated fire.

Options

Domain Object

An object that is used as a domain for the smoke simulation.

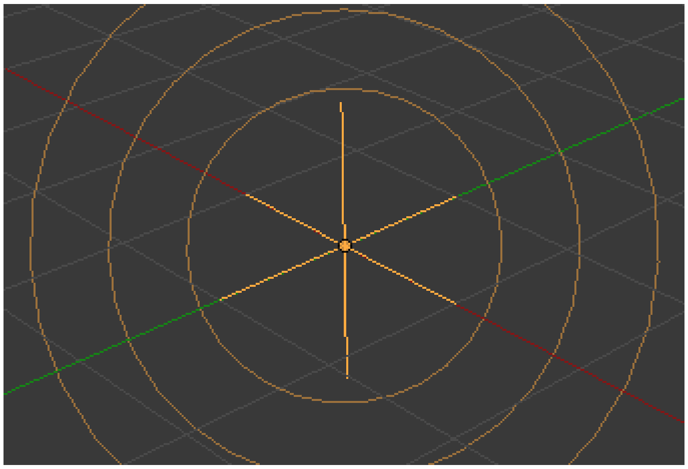
Apply Density

Adjust the force strength based on the smoke density.

Force

Reference

Panel::	*Physics* ▸ *Force Fields*
Type::	Force



Force force field visualization.

The *Force* field is the simplest of the fields. It gives a constant force towards (positive strength) or away from (negative strength) the object’s origin. Newtonian particles are attracted to a field with negative strength, and are blown away from a field with positive strength.

Example

Harmonic

Reference

Panel::	*Physics ▸ Force Fields*
Type::	Harmonic

In a *Harmonic* force field, the source of the force field is the zero point of a harmonic oscillator (spring, pendulum). If you set the *Damping* parameter to 1, the movement is stopped in the moment the object is reached. This force field is really special if you assign it to particles.

Options

Rest Length
  Controls the rest length of the harmonic force.

Multiple Springs
  Causes every point to be affected by multiple springs.

Normally every particle of the field system influences every particle of the target system. Not with *Harmonic*! Here every target particle is assigned to a field particle. So particles will move to the place of other particles, thus forming shapes.

Example

Lennard-Jones

Reference

Panel:: *Physics ▸ Force Fields*
Type:: Lennard-Jones

The *Lennard-Jones* force field is a very short range force with a behavior determined by the sizes of the effector and effected particle. At a distance smaller than the combined sizes, the field is very repulsive and after that distance it is attractive. It tries to keep the particles at an equilibrium distance from each other. Particles need to be at a close proximity to each other to be effected by this field at all.

Particles can have for example both a charge and a Lennard-Jones potential, which is probably something for the nuclear physicists among us.

Example

Magnetic

Reference

Panel::	*Physics* ▸ *Force Fields*
Type::	Magnetic

This field depends on the speed of the particles. It simulates the force of magnetism on magnetized objects.

Example

Texture

Reference

Panel::	*Physics ‣ Force Fields*
Type::	Texture

You can use a *Texture* force field to create an arbitrarily complicated force field, which force in the three directions is color coded. Red is coding for the X axis, green for the Y axis and blue for the Z axis (like the color of the coordinate axes in the 3D Viewport). A value of 0.5 means no force, a value larger than 0.5 acceleration in negative axis direction (like -Z), a value smaller than 0.5 acceleration in positive axis direction (like +Z).

Options

Texture Mode

This sets the way a force vector is derived from the texture.

Curl::	Calculates the force vector from the curl of the 3D-RGB texture (rotation of RGB vectors). This also works only with a color texture. It can be used for example to create a nice looking turbulence force with a color clouds texture with Perlin noise.
Gradient::	Calculates the force vector as the 3D gradient of the intensity (grayscale) of the texture. The gradient vector always points to the direction of increasing brightness.
RGB::	Uses the color components directly as the force vector components in the color encoded directions. You need an RGB texture for this, e.g. an image or a color ramp. So a *Blend* texture without a color ramp would not suffice.

Nabla

It is the offset used to calculate the partial derivatives needed for *Gradient* and *Curl* texture modes.

Use Coordinates

Uses the emitter object coordinates (and rotation & scale) as the texture space the particles use. Allows for moving force fields, that have their coordinates bound to the location coordinates of an object.

Root Texture Coordinates

This is useful for hair as it uses the texture force calculated for the particle root position for all parts of the hair strand.

2D

Disregards the particles Z coordinate and only uses particles X & Y as the texture coordinates.

Remember that only procedural texture are truly 3D.

Examples

- A single colored texture (0.5, 0.0, 0.5) creates a force in the direction of the positive Y axis, e.g. hair is orientated to the Y axis.
- A blend texture with color ramp can be used to created a force "plane". E.g. on the left side (0.5, 0.5, 0.5), on the right side (1.0, 0.5, 0.5) you have a force plane perpendicular to XY (i.e. parallel to Z). If you use an object for the coordinates, you can use the object to push particles around.
- An animated wood texture can be used to create a wave like motion.

Turbulence

Reference

Panel::	*Physics* ▸ *Force Fields*
Type::	Turbulence

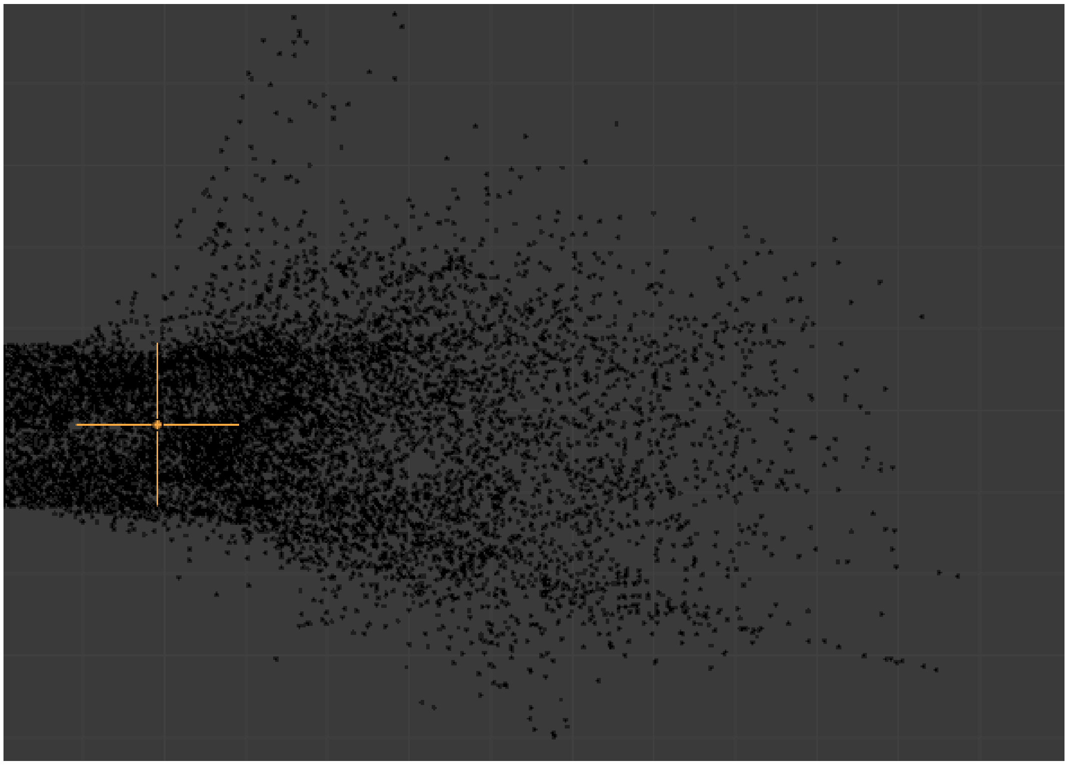
A *Turbulence* force field creates a random & chaotic 3D noise effect, similar to jets of water or geysers under the ocean.

Options

Size
: Indicates the scale of the noise.

Global
: Makes the size and strength of the noise relative to the world, instead of the object it is attached to.

Example

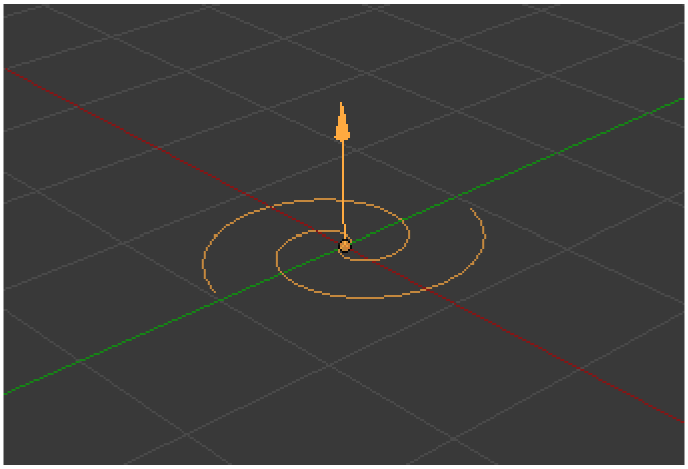


Turbulence force field affecting a particle system.

Vortex

Reference

Panel::	*Physics ‣ Force Fields*
Type::	Vortex



Vortex force field visualization.

The *Vortex* force field gives a spiraling force that twists the direction of points around the force object's local Z axis. This can be useful for making a swirling sink, or tornado, or kinks in particle hair.

Options

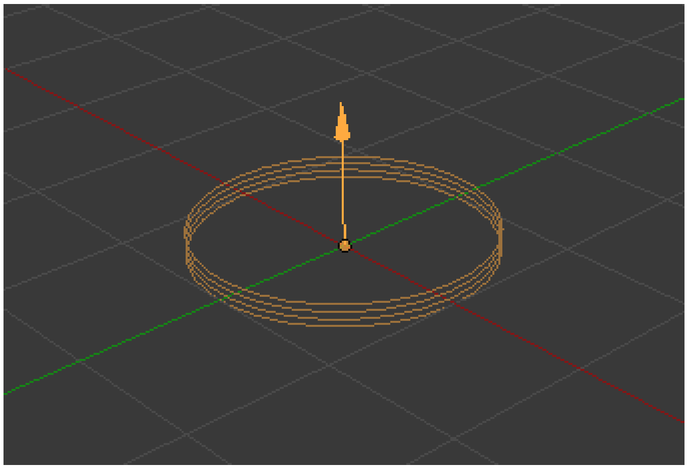
Inflow
  Inwards component of the vortex force.

Example

Wind

Reference

Panel::	*Physics* ▸ *Force Fields*
Type::	Wind



Wind force field visualization.

The *Wind* force field gives a constant force in a single direction, along the force object's local Z axis. The strength of the force is visualized by the spacing of the circles shown.

Example

Collision

Reference

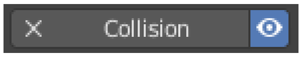
Mode:: Object Mode
Panel:: *Physics* ▸ *Collision*

Particles, Soft Bodies and Cloth objects may collide with mesh objects. Boids try to avoid *Collision* objects.

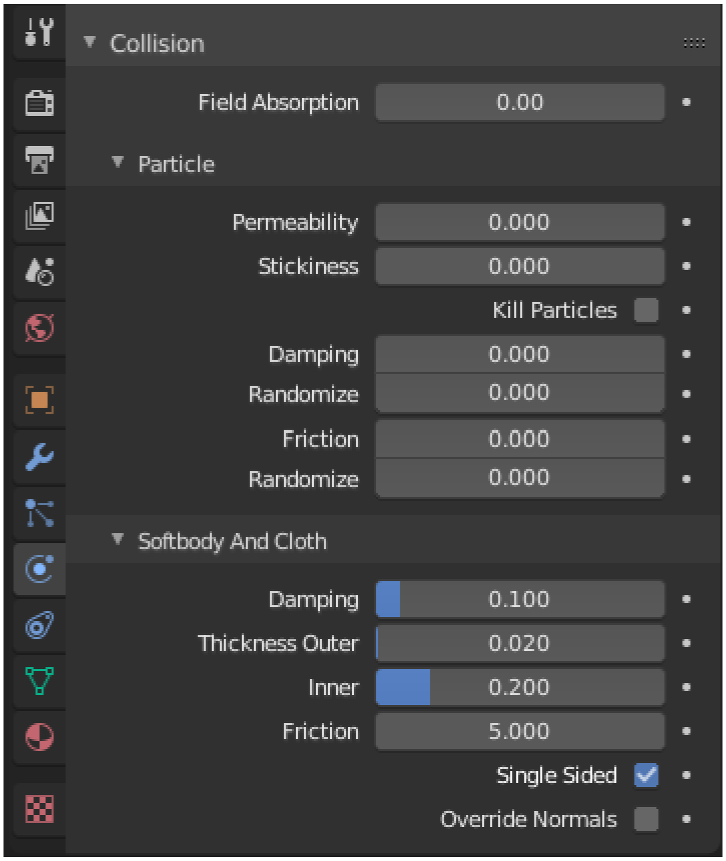
- You may limit the effect on particles to a group of objects (in the Field Weights panel).
- *Deflection* for soft body objects is difficult, they often penetrate the colliding objects.
- Hair particles ignore deflecting objects (but you can animate them as soft bodies which take deflection into account).

If you change the deflection settings for an object you have to recalculate the particle, soft body or cloth system by *Delete Bake*, this is not done automatically.

A collider object can be temporarily disabled via an animatable toggle to the right of the button that permanently activates or deactivates it.



Options



Collision

Field Absorption 0.00

Particle

Permeability 0.000

Stickiness 0.000

Kill Particles ☐

Damping 0.000

Randomize 0.000

Friction 0.000

Randomize 0.000

Softbody And Cloth

Damping 0.100

Thickness Outer 0.020

Inner 0.200

Friction 5.000

Single Sided ☑

Override Normals ☐

Collision panel.

Collision

Field Absorption

A deflector can also deflect effectors. You can specify some collision/deflector objects which deflect a specific portion of the effector force using the *Field Absorption* value. 100% absorption results in no force getting through the collision/deflector object at all. If you have three collision object behind each other with e.g. 10%, 43% and 3%, the absorption ends up at around 50% \(100 × (1 - 0.1) × (1 - 0.43) × (1 - 0.03)\).

Particle

Permeability

Fraction of particles passing through the mesh.

Stickiness
  How much particles stick to the object.

Kill Particles
  Deletes Particles upon impact.

Damping
  Damping during a collision (independent of the velocity of the particles).

  Randomize
    Random variation of damping.

Friction
  Friction during movements along the surface.

  Randomize
    Random variation of friction.

Soft Body and Cloth

It is also important to note that this collision panel is used to tell all simulations that this object is to participate in colliding/deflecting other objects on a shared layer (particles, soft bodies, and cloth).

Note

The object’s shape deforms the cloth, so the cloth simulation must be inputted the “true” shape of that mesh object at that frame. This true shape is the basis shape as modified by shape keys or armatures. Therefore, the Collision Modifier must be **after** any of those. The image to the right shows the *Modifiers* panel for the Character mesh object (not the cloth object).

Damping

Damping during a collision. The amount of bounce that the surfaces will have.

- 0.0 - No damping, soft bodies will have a maximum bounciness.
- 1.0 - Maximum damping, soft bodies will not bounce at all.

Thickness

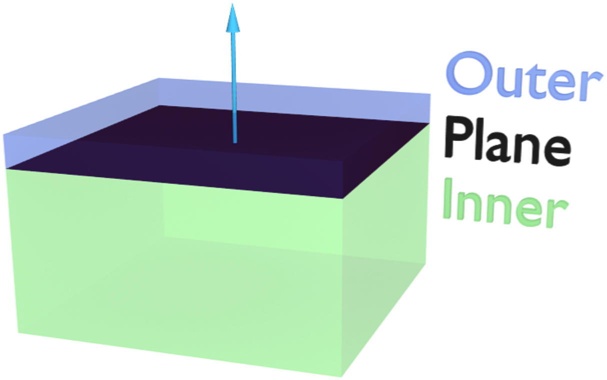
A padding distance is added to the inside and outside of each face, to help to prevent intersections. The soft body will come to rest at this distance away from the face of the colliding object. Outside and inside is defined by the face normal, depicted as blue arrow in Fig. A soft body vertex colliding with a plane..

Outer

Size of the outer collision zone.

Inner

Size of the inner collision zone (padding distance).



A soft body vertex colliding with a plane.

Friction

A coefficient for how slippery the cloth is when it collides with itself. For example, silk has a lower coefficient of friction than cotton.

Single Sided

When enabled, the collider is considered to represent the boundary of a solid object rather than a thin surface, and ejects intersecting cloth in the direction of its normal.

Override Normals
When enabled, cloth collision impulses act in the direction of the collider normals.

Note

Soft body collisions are difficult to get perfect. If one of the objects move too fast, the soft body will penetrate the mesh. See also the section about Soft Bodies.

Examples

Deflected particles.

Here is a *Meta* object, using Instancing Vertices to a particle system emitting downwards, and deflected by a mesh cube.

Hints

- Make sure that the normals of the mesh surface are facing towards the particles/points for correct deflection. Negative scales on the object can have a similar effect. Make sure to recalculate the normals after applying the scale.
- Hair particles react directly to force fields, so if you use a force field with a short range you do not need necessarily collision.
- Hair particles avoid their emitting mesh if you edit them in *Particle Edit Mode*. So you can at least model the hair with collision.

Baking Physics Simulations

Baking refers to the act of storing or caching the results of a calculation. The result of a simulation is automatically cached to memory when the animation is played, so that the next time it runs, it can be replayed more quickly by reading the results from the memory.

If you bake the simulation the cache is protected, and you will be unable to change the simulation settings until you clear the baked frames by clicking *Delete Bake*.

It is generally recommended to bake your physics simulations before rendering. Aside from no longer needing to go through the time-consuming process of simulating again, baking can help prevent potential glitches and ensure that the outcome of the simulation remains exactly the same every time.

Note

Most physics simulators in Blender use a similar system, but not all have exactly the same settings available. All the settings are covered here, but individual physics types may not provide all these options.

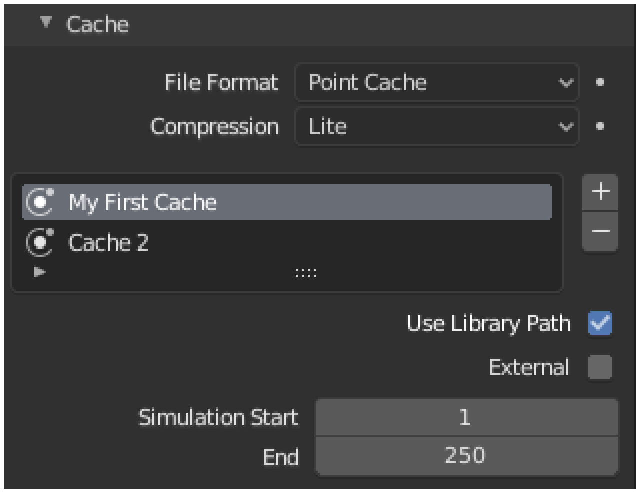
Options

Caches List

Blender allows for storing and managing multiple caches at once for the same physics object. You can manage the caches with this list view. Double-click the cache entry to give it a name.

Each cache can have a name. Double-click the cache entry to give it a name. If this name is given, any disk cache will be stored in files starting with that name. For example, a cache named `MyCache` will be stored in `MyCache_xxxxxx_yy.bphys`.

If the cache does not have a name (which is the default), the filename of the cache will depend on the object it is attached to, although this is not immediately obvious. For example, a cache on an object `Cube` will be stored in `43756265_xxxxxx_yy.bphys`, where `43756265` is determined by the object name.



File Format: Point Cache
Compression: Lite

- My First Cache
- Cache 2

Use Library Path ☑
External ☐
Simulation Start: 1
End: 250

Two different caches stored simultaneously.

Warning

When there are multiple caches on one object, **always specify a Cache Name**. As described above, the filename of an unnamed cache is determined by the name of the object it is attached to. As a result, an object with multiple physics systems that all have an unnamed cache will cause conflict and **can result in losing cache files**.

External

Allows you to read the cache from a drive using a user-specified file path.

Note

The cache name in *Caches List* and the *Index* has to exactly match the external cache files name in order to work. The cache files name format is `name_frame_index.bphys`.

Index

The index number of cache files. (The last two digits of the files name.)

Path

Select the directory path to the cache files.

Disk Cache

The cache of a baked simulation will be stored inside the blend-file when you save it. When *Disk Cache* is checked, Blender will save the cache separately to the drive in a folder named `blendcache_[filename]` alongside the blend-file. (The blend-file must be saved first.)

Note

When using Library Overrides, data-blocks only support *Disk Cache* storage.

Use Library Path

Share the disk cache when the physics object is linked into another blend-file. When this option is enabled, linked versions of the object will reference the same disk cache. Otherwise linked versions of the object will use independent caches.

Compression

The compression level for cache files. Some physics caches can be very large, Blender can compress these caches in order to save space.

None

Do not compress the cache.

Light

Compression will optimize the speed of compressing/decompressing operations over file size.

Heavy

Compression will result in smaller cache files more than *Light*, however, requires more CPU time to compress/decompress.

Start

Frame on which to start the simulation.

End

Frame on which to stop the simulation.

Note

The simulation is only calculated for positive frames in between the *Start* and *End* frames of the *Cache* panel, whether you bake or not. So if you want a simulation that is longer than the default frame range you have to change the *End* frame.

Cache Step

Interval for storing simulation data.

Note

Some physics systems (such as particles) allow for positions to be stored only on every nth frame, letting the positions for in-between frames be interpolated. Using a cache step greater than one will result in a smaller cache, but the result may differ from the original simulation.

Baking

Bake

Start baking. Blender will become unresponsive during most baking operations. The cursor will display as a number representing the progress of the baking. You need to be in Object Mode to bake.

Delete Bake

Mark the baked cache as temporary. The data will still exist, but will be removed with the next object modification and frame change. This button is only available when the physics system has been baked.

Calculate to Frame

Bake only up to the current frame. Limited by *End* frame set in the cache settings.

Current Cache to Bake

Store any temporarily cached simulation data as a bake. Note that playing the animation will try to simulate any visible physics simulations. Depending on the physics type, this data may be temporarily cached. Normally such temporary caches are cleared when an object or setting is modified, but converting it to a bake will “save” it.

Bake All Dynamics

Bake all physics systems in the scene, even those of different types. Useful for baking complex setups involving interactions between different physics types.

See Bake.

Delete All Bakes

Delete bakes of all physics systems in the scene, even those of different types.

See Delete Bake for more information.

Update All to Frame

Bake all physics systems in the scene to the current frame.

See Calculate To Frame.

Simulation Nodes

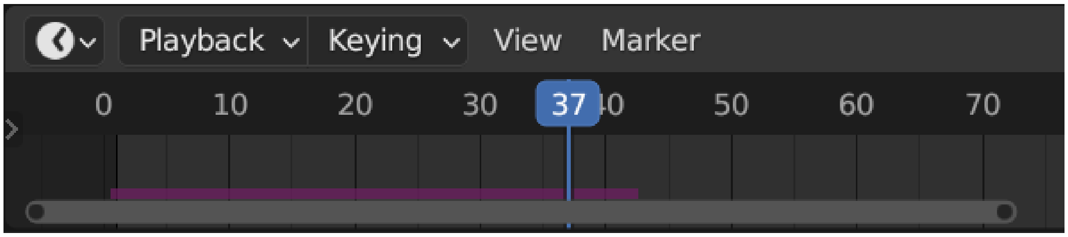
Through the use of Simulation Zones, Geometry Nodes can be used to create custom physic simulations through nodes. Simulation zones allow the result of one frame to influence the next one. That way even a set of simple rules can lead to complex results, with the passing of time. The most common type of them is physics simulation, with specific solvers for physical phenomena.

See also

Read more about Simulation Zones

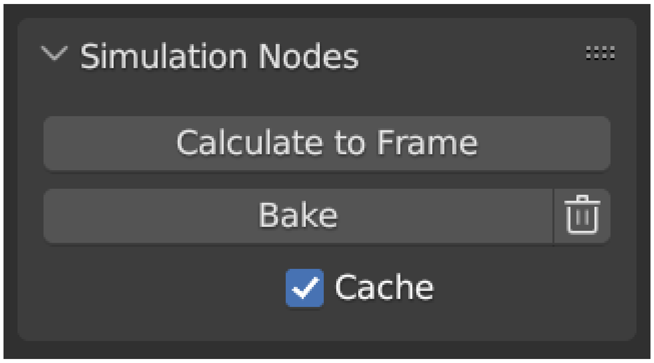
Baking

The simulation is automatically cached during playback. The valid cache can be seen as a strong yellow line in the timeline editor. This allows for animators to quickly inspect all the previous frames of a simulation.



Cached frames in the Timeline.

When the result is ready to be sent to a render-farm, it can be baked to disk. This allows for the simulation to be rendered in a non-sequential order.



☑ Cache

Simulation and Physics, Geometry Nodes user interface.

Note

Baking the simulation will bake all the simulations in all modifiers for the selected objects.

Calculate to Frame

Calculate simulations in geometry nodes modifiers from the start to current frame.

Bake

Bake simulations in geometry nodes modifiers. In order to bake the simulation, the blend-file must be saved to your computer. The location the file is saved determines where the baked data is also saved. The directory the baked data is saved to can be changed per modifier in the Internal Dependencies.

Delete Cached Simulation

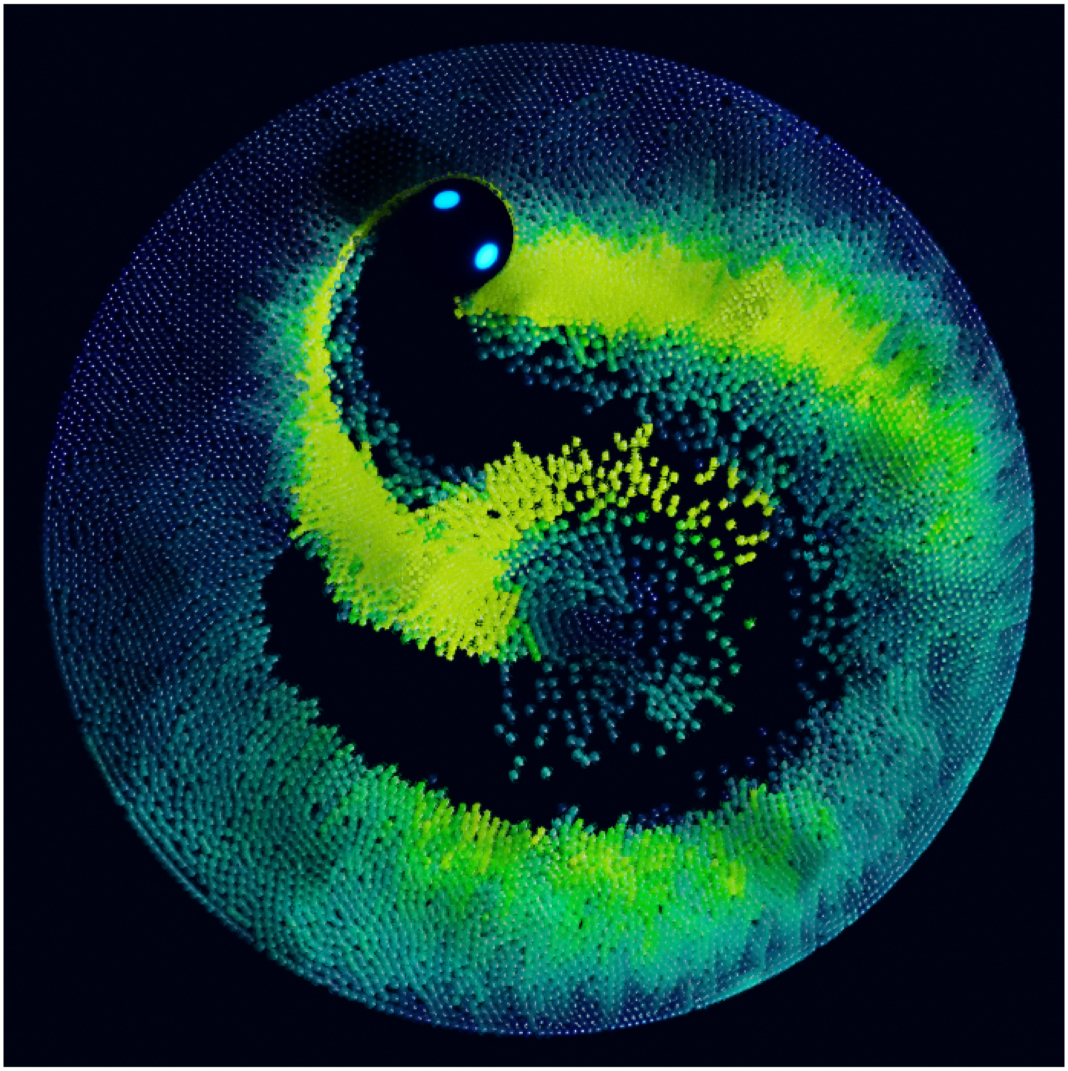
Delete cached/baked simulations in geometry nodes modifiers

Cache

For the cases where the current frame is the only one relevant, users can opt-out of caching the results to save memory.

Examples

Combined with the Index of Nearest, this can be used for a number of sphere-based simulations.



Index of Nearest sample file CC-BY Sean Christofferson.

Rendering

- Introduction
- EEVEE
  - Introduction
  - Render Settings
  - Scene Settings
  - World Settings
  - Object Settings
  - Materials
  - Light Settings
  - Light Probes
  - Limitations
- Cycles
  - Introduction
  - Render Settings
  - World Settings
  - Object Settings
  - Material Settings
  - Light Settings
  - GPU Rendering
  - Experimental Features
  - Render Baking
  - Optimizing Renders
- Workbench
  - Introduction
  - Performance
  - Sampling
  - Lighting
  - Color
  - Options
  - Grease Pencil
  - Viewport Display
- Cameras
  - Properties

- Lights
    - Light Objects
    - Light Linking
    - World Environment
- Materials
    - Introduction
    - Components
    - Assignment
    - Preview
    - Settings
    - Line Art
    - Legacy Textures
- Shader Nodes
    - Introduction
    - Input
    - Output
    - Shader
    - Texture
    - Color
    - Vector
    - Converter
    - Group
    - Open Shading Language
- Color Management
    - Workflow
    - Render Settings
    - Image Files
    - OpenColorIO Configuration
    - Default OpenColorIO Configuration
- Freestyle
    - Introduction
    - Render Properties
    - View Layer Properties
    - Material Properties
    - Python Scripting
- Layers & Passes
    - Introduction

  - View Layer
  - Passes
- Render Output
  - Introduction
  - Output Properties
  - Audio Rendering
  - Rendering Animations
  - Animation Player

Introduction

Rendering is the process of turning a 3D scene into a 2D image. Blender includes three render engines with different strengths:

- EEVEE is a physically based realtime renderer.
- Cycles is a physically based path tracer.
- Workbench is designed for layout, modeling and previews.

More renderers from third-party developers are available as add-ons. Each renderer has its own render settings to control render quality and performance.

What the render looks like is defined by cameras, lights and materials. These are shared between EEVEE and Cycles, however some features are only supported in one or the other.

Renders can be split up into layers and passes, which can then be composited together for creative control, or to combine with real footage. Freestyle can be used to add non-photorealistic line rendering.

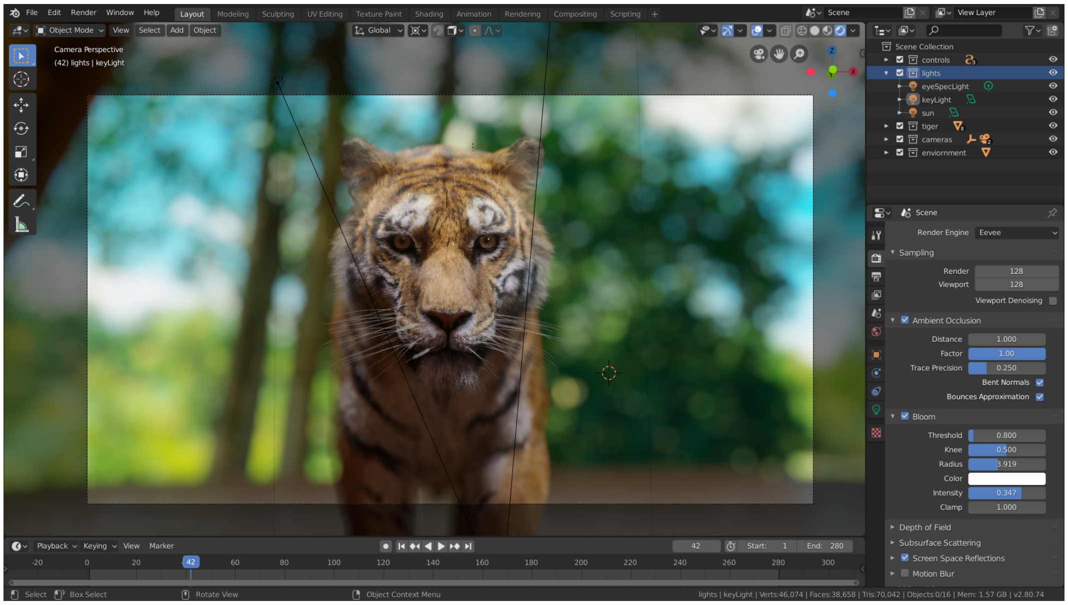
Blender supports interactive 3D viewport rendering for all render engines, for quick iteration on lighting and shading. Once this is done, the final quality image or animation can be rendered and output.

EEVEE

- Introduction
- Render Settings
    - Sampling
    - Clamping
    - Raytracing
    - Volumes
    - Curves
    - Depth of Field
    - Motion Blur
    - Film
    - Performance
    - Grease Pencil
- Scene Settings
- World Settings
- Object Settings
    - Object Properties
- Materials
- Light Settings
- Light Probes
    - Sphere
    - Plane
    - Volume
- Limitations
    - Limitations
    - Supported Nodes

Introduction

EEVEE is Blender’s realtime render engine focused on speed and interactivity while achieving the goal of rendering PBR materials. EEVEE can be used interactively in the 3D Viewport but also produce high quality final renders.


File
Edit
Render
Window
Help
Layout
Modeling
Sculpting
UV Editing
Texture Paint
Shading
Animation
Rendering
Compositing
Scripting
Scene
View Layer
Object Mode
View
Select
Add
Object
Global
Camera Perspective
(42) lights | keyLight
Scene Collection
controls
lights
eyeSpecLight
keyLight
sun
tiger
cameras
enviornment
Scene
Playback
Keying
View
Marker
-20
0
20
42
60
80
100
120
140
160
180
200
220
240
260
280
300
Select
Box Select
Rotate View
Object Context Menu

Render Engine: Eevee

Sampling
Render: 128
Viewport: 128
Viewport Denoising ☐

☑ Ambient Occlusion
Distance: 1.000
Factor: 1.00
Trace Precision: 0.250
Bent Normals ☑
Bounces Approximation ☑

☑ Bloom
Threshold: 0.800
Knee: 0.500
Radius: 3.919
Color
Intensity: 0.347
Clamp: 1.000

Depth of Field
Subsurface Scattering
☑ Screen Space Reflections
☐ Motion Blur

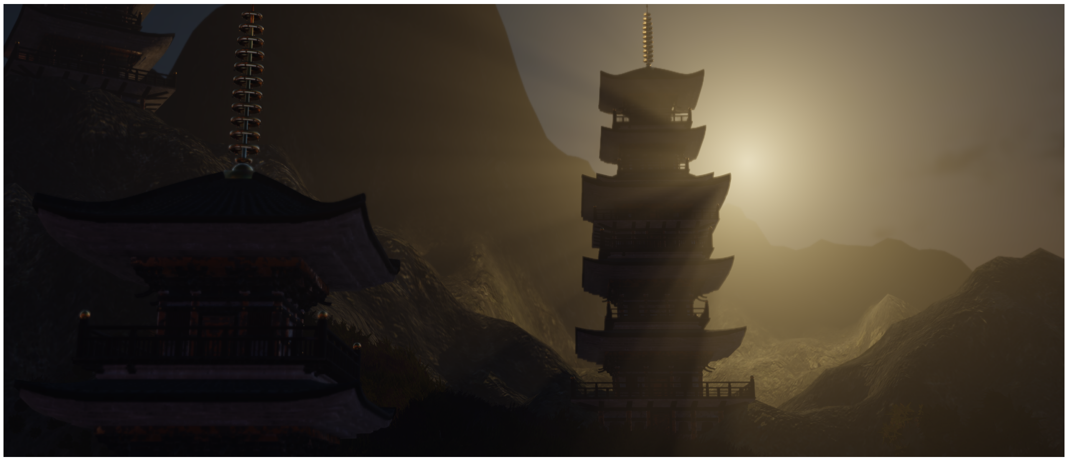
Frame: 42 Start: 1 End: 280

lights | keyLight | Verts:46,074 | Faces:38,658 | Tris:70,042 | Objects:0/16 | Mem: 1.57 GB | v2.80.74

EEVEE in the 3D Viewport – “Tiger” by Daniel Bystedt.

EEVEE materials are created using the same shader nodes as Cycles, making it easy to render existing scenes. For Cycles users, this makes EEVEE work great for previewing materials in realtime.

EEVEE is based on rasterization and is not a path tracer. Instead of computing each ray of light, rasterization determines what surface is visible from the camera. It then estimates the way light interacts with these surfaces and materials using numerous algorithms. While EEVEE is designed to use PBR principles, it is not perfect and Cycles will always provide more physically accurate renders. For these reasons, EEVEE has a set of limitations.



EEVEE final render – “Temple” by Dominik Graf.

Render Settings

- Sampling
- Clamping
- Raytracing
- Volumes
- Curves
- Depth of Field
- Motion Blur
- Film
- Performance
- Grease Pencil

Sampling

EEVEE uses a process called Temporal Anti-Aliasing (TAA) which reduces Aliasing. TAA is sample based so the more samples the more aliasing is reduced at the cost of performance.

Reference	
Panel::	*Render ▸ Sampling*

Viewport

Samples

The number of samples to use in the 3D Viewport. When setting this to zero the viewport will be resampled continuously.

Temporal Reprojection

Reduces noise while moving the viewport or during animation playback. Can leave some ghosting.

Jittered Shadows

Enable jittered shadows on the viewport. Jittered shadows are always enabled for final renders. This also affects shadows casted by transparent shadows.

Render

Samples

The number of samples to use in the final render.

Shadows

Rays

Number of rays to trace for each light. Higher values reduces the noise caused by random shadow sampling.

Steps

Number of shadow map sample per shadow ray. Higher step count results in softer shadows but have a higher cost.

Volumetric Shadows

Approximate light absorption of the surrounding volume objects. This makes the volumes more opaque to light. This is a very computationally expensive option and has limitations.

Steps

Number of steps to compute volumetric shadowing.

See also

Volume Limitations.

Resolution

Resolution percentage of shadow maps.

Advanced

Light Threshold

Minimum light intensity for a light to contribute to the lighting. Used to compute the distance at which to cut-off lights influence. Lower values improve performance.

See also

Custom Distance overrides this setting.

Clamping

Reference

Panel:: *Render ‣ Clamping*

Surface

Direct Light

This option limits the maximum light intensity a surface can reflect. It reduces Aliasing noise and Fireflies at the cost of accuracy. Setting this option to 0.0 disables clamping altogether. Lower values have a greater effect on the resulting image than higher values.

Indirect Light

Similar to **Direct Light** but limits the maximum light intensity reflected using ray-tracing and light-probes.

Note

These options provide a way to limit Fireflies and Aliasing of highly reflective surfaces and dense volumes. However, note that as you clamp out such values, other bright lights will be dimmed as well.

Care must be taken when using this setting to find a balance between mitigating fireflies and losing intentionally bright parts.

Volume

Direct Light

The same as *Surface Direct Light* but for volume direct lighting.

Indirect Light

The same as *Surface Direct Light* but for volume indirect lighting.

Raytracing

Reference

Panel::	*Render* ▸ *Raytracing*

The ray-tracing pipeline goal is to increase the accuracy of surface indirect lighting. This is done by generating ray from each BSDF and finding their intersection with the scene individually.

When disabled, it is replaced by a faster pipeline that uses pre-filtered light-probes. This fallback mode offers a more visually stable and optimized alternative when visual fidelity is not the primary goal.

See also

Limitations.

Method

Determine the tracing method used to find scene-ray intersections and indirect lighting.

Light Probe::	Use light-probe spheres and planes to find scene intersection. This option has the lowest tracing cost but relies on manually placed light-probes.
Screen-Trace::	Trace ray against the screen depth buffer. Fallback to light-probes if ray exits the view.

Resolution

Resolution at which the ray-tracing is performed. Lower options will be faster and use less memory but will produce blurrier results.

Screen Tracing

These settings control the behavior of the screen space ray-tracing. They are only visible if *Screen-Trace* is the active tracing *Method*.

Precision
: Higher values increase precision of the screen space ray-tracing but lower the maximum trace distance. Increased precision also increases performance cost.

Thickness
: How thick to consider the pixels of the depth buffer during the tracing. Higher values will stretch the reflections and add flickering. Lower values may make the ray miss surfaces.

Denoising

Denoising can be enabled to reduce the amount of noise from the raw ray-traced output. This can help image stability but will also over-blur the final ray-traced output.

Spatial Reuse
: Reuse the rays from neighbor pixels. Can introduce some light leaks across surfaces.

Temporal Accumulation
: Accumulate samples by re-projecting the last ray tracing results. This removes Fireflies but also introduces color bias. Useful for viewport temporal stability or making renders converge faster.

Bilateral Filter
: Blur the resolved ray-traced output using a bilateral filter.

Fast GI Approximation

Fast GI Approximation is a fallback to the ray-tracing pipeline for BSDF with high roughness. It produces a less noisy output and captures bounce lighting more efficiently than individually traced rays.

This is currently implemented as a screen space effect and will inherit all associated limitations.

Threshold

Maximum roughness a BSDF can have to use ray-tracing. BSDFs with higher roughness will progressively use the *Fast GI Approximation*. A value of 1 will raytrace every surfaces and disable the Fast GI.

Method

Determine the method used to compute the fast GI approximation.

Ambient Occlusion::

Use scene intersections to shadow the distant lighting from light-probes. This is the fastest option.

Global Illumination::

Compute global illumination taking into account light bouncing off surrounding objects.

Resolution

Resolution at which the fast GI is computed. Lower options will be faster and use less memory but will produce blurrier results.

Rays

Number of GI rays per pixel at the specified *Resolution*. Higher values will reduce noise.

Steps

Number of screen samples per GI ray. Higher values will reduce the noise amount and increase the quality.

Tip

With a higher step count, there is less chance to miss other surfaces that could reflect or block the light. This means that the Fast GI *Thickness* parameters can be tweaked to lower values without losing too much light bounce energy.

Precision

Higher values increase the precision of the scene intersections with the GI rays. Increased precision also increases performance cost.

Distance

If non-zero, the maximum distance at which other surfaces will contribute to the fast GI approximation.

Thickness Near

Geometric thickness of the surfaces when computing fast GI and ambient occlusion. Reduces light leaking and missing contact occlusion. The effectiveness decreases proportionally to the distance from the shading point, following the inverse square law.

Far

Angular thickness of the surfaces when computing fast GI and ambient occlusion. Reduces energy loss and missing occlusion of far geometry. Higher values will make the very thin objects block or reflect too much light.

Bias

Bias the shading normal to reduce self intersection artifacts.

Volumes

Reference

Panel:: *Properties* ▸ *Render* ▸ *Volumes*

EEVEE simulates volumetric scattering by evaluating all volume objects inside the view frustum.

To achieve this, EEVEE uses several 3D textures which have a high video memory usage. The texture dimensions can be tweaked using the *Resolution* and *Steps* parameters.

Resolution
: Controls the quality of the volumetric effects. Lower resolution increases video memory usage and quality.

Steps
: Number of steps to compute volumetric effects. Higher count increases video memory usage and quality. These samples are distributed along the view depth (view Z axis).

Distribution
: Blend between linear and exponential sample distribution. Higher values put more samples near the camera.

Max Depth
: Maximum surface intersection count used by accurate volume intersection method. Will create artifacts if it is exceeded.

Custom Range

When working with volume objects, EEVEE automatically computes the best depth range where to compute the volume sampling and lighting. In certain situations, this isn’t enough and produces sub-optimal sampling which increases noise. This is particularly the case when using a volume shader inside the *World* or when working with large number of volume objects. The custom depth range can be enabled to restrict the computation of volumes to a certain range along the camera depth and thus increase precision.

Start
    Start distance of the volumetric effect.

End
    End distance of the volumetric effect.

See also

Limitations.

Curves

Reference

Panel::	*Render ‣ Curves*

Shape

Strand::	Render curves as a thin strand roughly a pixel wide. Curve diameter parameters are ignored with this setting.
Strip::	Render curves as a flat ribbon with rounded normals.

Additional Subdivisions

Additional subdivisions to be applied on top of the curve resolution set in the hair system settings. Increasing this value will smooth out the curves of the strands.

Depth of Field

To render a scene, EEVEE uses a pinhole camera model which produces a perfectly focused image of the scene. For an enhanced realism, EEVEE can simulate the optical Depth of Field using a post-process filter, and a sample-based method. The optical settings are located in the camera settings properties. Whereas the quality of the effect can be controlled by the settings found in the present section.

Note

In the 3D Viewport, depth of field only works while in Camera View.

The post-process method is computed in two passes. The first pass is using a blur that fails to produce quality bokeh for highlights but works for the general case. Followed by a second pass which is sprite-based and improves only the quality of very bright highlights. That is because it is too slow to be applied on every part of the image. So it just includes very bright isolated parts of the image such that are different from their surroundings. Which pixels are being processed by second pass can be control with the *Sprite Threshold* and *Neighbor Rejection* options.

Secondly the sample-based method works by randomizing the camera position for every sample. It is more accurate but needs many samples to achieve a smooth result. Accordingly the post-process blurring radius is scaled down to remove undersampling. Yet some scenes might still need more post-process blur in order to remove the noticeable sample pattern. This is exactly what the *Overblur* option does, but it will also reduce the bokeh shape sharpness.

Reference

Panel::	*Render* ▸ *Depth of Field*

Max Size

Maximum size in pixels of the depth of field post-process effect (lower is faster). A value of 0 will disable the post-process effect but not the sample-based method.

Sprite Threshold

Minimum brightness a pixel needs to have to be considered by the sprite-based depth of field. Higher values will improve the performance but will also reduce the quality of highlights. Brightness is in the scene's referred color space.

Neighbor Rejection

Maximum intensity to consider when doing sprite neighborhood rejection. This should be set to a brightness value above which there is small visual differences to be noticeable after color management. Lower values will improve the performance but will also reduce the quality of highlights. Brightness is in the scene's referred color space.

Jitter Camera

Randomize the camera position for every scene render sample to increase precision. Enabling this option can change the scene's actual sample count.

Note

Be aware that the actual sample count can grow quite rapidly.

Hint

The actual number of samples is computed by the following formula:

$$sample\_count = (ring\_count^{2} + ring\_count) * 3 + 1$$

where $ring\_count$ is the number of ring in the hexaweb pattern. The $ring\_count$ is chosen so that the entire pattern contains at least the number of samples set in the Render Settings.

Over-blur

Scales the post-process depth of field radius to reduce artifacts. Higher values will soften the bokeh shape.

See also

Limitations.

Motion Blur

Reference

Panel::	*Render ‣ Motion Blur*

Blender's animations are by default rendered as a sequence of *perfectly still* images. While great for stop-motion and time-lapses, this is unrealistic, since fast-moving objects do appear to be blurred in the direction of motion, both in a movie frame and in a photograph from a real-world camera.

Note

Motion blur is only visible in the viewport during animation playback and uses a simpler algorithm than final render. Same thing applies to Viewport Renders.

Position

Controls at what point the shutter opens in relation to the current frame.

Start on Frame::	Shutter is starting to open at the current frame.
Center on Frame::	Shutter is fully opened at the current frame.
End on Frame::	Shutter is fully closed at the current frame.

Shutter

Time (in frames) taken between shutter open and close.

Bleeding Bias

Used by the post-process blur to avoid blurring the background over the foreground. Lower values will reduce background bleeding onto foreground elements.

Max Blur

Max Blur is intended to act as an optimization tool by limiting the number of pixels across which the blur is calculated.

Steps

This controls the number of steps used by the accumulation blur and thus its accuracy. More steps means longer render time.

Note

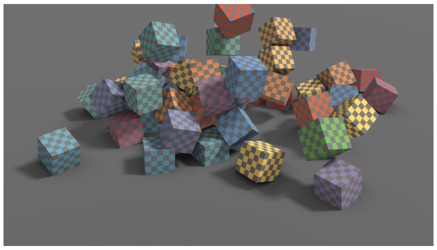
When using multiple time steps, the render sample count is rounded up to the next multiple of steps to ensure even distribution of samples across steps.

EEVEE splits the render into multiple time steps and accumulates the result which is known as Accumulation Motion Blur. This technique is precise but requires many steps for clean gradients. This is used in combination with the post-process blur to handle the inter-step gaps. Each step corresponds to a full scene re-evaluation and can add a lot of overhead to the render time. By adding more steps you can also reduce the *Max Blur* options because the post-process blur has to cover a smaller distance.

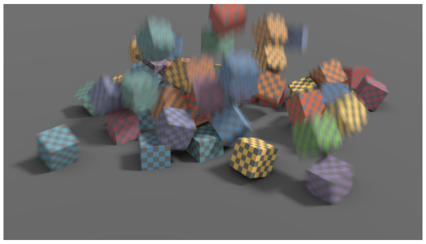
Shutter Curve

Use a custom shutter curve.

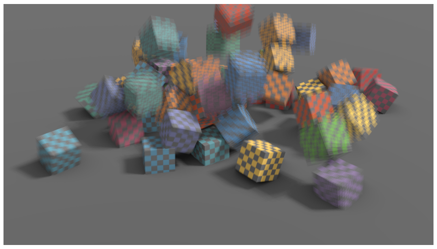
Example



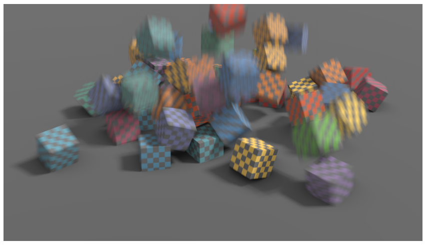
No motion blur.



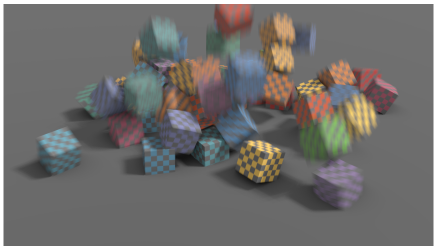
Only post-process blur.



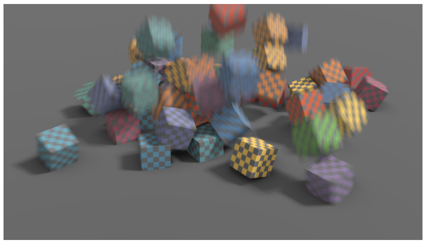
4 time steps without post-process blur.



4 time steps with post-process blur.



32 time steps without post-process blur.



32 time steps with post-process blur.

Film

Filter Size

Due to limited resolution of images and computer screens, pixel filters are needed to avoid Aliasing. This is achieved by slightly blurring the image to soften edges.

This Setting controls how much the image is softened; lower values give more crisp renders, higher values are softer and reduce aliasing.

Transparent

Render the background transparent, for compositing the image over another background after rendering.

Overscan

Percentage of the render size to add to the internal render buffer. This will have a serious impact on performance but can fix render glitches around the perimeter of the rendered image.

Performance

Reference

Panel:: *Properties* ▸ *Render* ▸ *Performance*

High Quality Normals

Uses higher precision normals and tangents which can improve visual quality for dense meshes with high frequency textures at the cost of memory.

Memory

Shadow Pool

A bigger pool size allows for more shadows in the scene but might not fit into GPU memory and decreases performance. Increasing the size might fix the *Shadow buffer full* error.

See also

Shadow documentation

Light Probes Volume Pool

A bigger pool size allows for more irradiance grids in the scene but might not fit into GPU memory and decreases performance.

Viewport

Pixel Size

Option to control the resolution for viewport rendering. Allows you to speed up viewport rendering, which is especially useful for displays with high DPI.

Grease Pencil

Reference

Panel:: *Render ‣ Grease Pencil*

This panel is comprised of settings to control the rendering of Grease Pencil Lines.

Anti-Aliasing Threshold
    Threshold for the edge detection algorithm used to correct aliasing, higher values might over blur some part of the image.

Scene Settings

Light Probes

Light Probe Spheres Resolution

Defines the resolution of every light probe sphere in the scene.

World Settings

The world environment can emit light, ranging from a single solid color to arbitrary textures.

In EEVEE, the world lighting contribution is stored into an internal Light Probe. This makes the lighting less precise than Cycles.

Mist Pass

Reference

Panel::	*World ▸ Mist Pass*

Note

The mist pass must be enabled in the View Layer tab of the Properties Editor before the settings below are available in the World tab.

Mist can greatly enhance the illusion of depth in your rendering. To create mist, Blender generates a render layer with a depth map ranging between 0.0 and 1.0 that can be used in the Compositor to generate a mist effect.

Start
: The distance from the camera at which the mist starts to fade in.

Depth
: The distance from *Start* of the mist, that it fades in over. Objects further from the camera than *Start* + *Depth* are completely hidden by the mist.

Falloff
: The curve function that controls the rate of change of the mist’s strength further and further into the distance.

Quadratic::	Uses the same calculation as light falloff ($1\over{x^2}$) and provides the smoothest transition from transparent (0.0) to opaque (1.0).
Linear::	Has a steeper start than quadratic ($1\over{x}$).
Inverse Quadratic::	Has the steepest start ($1\over{\sqrt{x}}$) and approaches 1.0 faster than the other two functions.

Tip

A visualization can be activated in the *Camera* ▸ *Viewport Display* panel.

Mist example (blend-file).

Settings

Reference

Panel:: *World* ▸ *Light Probe*

Light Probe

Resolution

The resolution used to store the light from the world. This is equivalent to the resolution for light probe spheres.

See also

Light Probe Sphere.

Sun

EEVEE can separate the light from intense light sources (e.g. a sun from an outdoor HDRI) and replace them with a sun light. This increases the quality of the lighting as the internal light probes alone cannot reproduce this type of lighting with enough precision.

Threshold

If non-zero, the maximum value for world contribution that will be recorded inside the world light probe. The excess contribution is converted to a sun light. This reduces the light bleeding caused by very bright light sources. A value of zero will disable this feature and all lighting will be stored inside the internal light probes.

Angle

Angular diameter of the extracted sun light as seen from the Earth.

Use Shadow

Enable shadow casting on the extracted sun light.

See also

The shadow properties control the extracted sun shadows. They are exactly the same as for a sun light object. Light Properties.

Object Settings

Settings for objects and object data.

- Object Properties
    - Shading
    - Visibility

Object Properties

Shading

Reference	
Panel::	*Properties* ▸ *Object Properties* ▸ *Shading*

Light Linking

Limit light influence to specified objects, with Light Linking.

Receiver Collection
Collection of objects that will receive light emitted from the object.

Shadow Linking

Limit shadows to specified objects, with Light Linking.

Shadow Blocker Collection
Collection of objects that will act as shadow blockers for light emitted from the object.

Visibility

Reference	
Panel::	*Object Properties* ▸ *Visibility*

Ray Visibility

Objects can be set to be invisible to particular ray types. This can be used, for example, to make an emitting mesh invisible to camera rays. For instanced objects, visibility is inherited; if the parent object is hidden for some ray types, the children will be hidden for these too.

In terms of performance, using these options is more efficient that using a shader node setup that achieves the same effect.

Camera
: Makes the object visible to the Camera; this includes the viewport's perspective in viewport rendering.

Shadow
: Enables the object to cast shadows. The object will not be capture inside the shadow maps.

Light Probes

Objects can be set to not be captured by certain light probe. This can be used, for example, to avoid animated object being recorded into static light probes. For instanced objects, visibility is inherited; if the parent object is hidden for some ray types, the children will be hidden for these too.

Volume
: Makes the object visible during light probe volumes baking.

Sphere
: Makes the object visible during light probe sphere capture.

Plane
: Makes the object visible during light probe plane capture.

Materials

See also

While EEVEE shares the same material node system as Cycles, not all features are supported. See Shader nodes limitations.

Thickness

Reference

Panel:: *Properties ‣ Material ‣ Thickness*

Used to approximate the inner geometry structure of the object without heavy computation. This is currently used for Subsurface Scattering, Translucent BSDF, Refraction BSDF, and the nodes containing these effects.

If no value is plugged into the output node, a default thickness based on the smallest dimension of the object is used. If a value is connected it will be used as object space thickness (i.e. scaled by object transform). A value of zero will disable the thickness approximation and treat the object as having only one interface.

This output is only used by the EEVEE render engine.

Note

- The thickness is used to skip the inner part of the object.
- Refraction will not refract objects inside the thickness distance.

- Shadow casting object will not cast shadow within the thickness distance.

Tip

- For large or compound meshes (e.g. vegetation), the thickness should be set to the thickness of individual parts (e.g. leaves, grass blades).
- Thickness can be baked to textures or custom attributes for more accurate result.

See also

Thickness Mode – controls how the thickness value is used.

Material Settings

Reference

Panel:: *Properties* ▸ *Material* ▸ *Settings*

Pass Index

Index number for the *Material Index* render pass. This can be used to give a mask to a material which then can be read with the ID Mask Node in the Compositor.

Note

Volume Objects do not support the pass index.

Surface

Backface Culling

Backface Culling hides the back side of faces. This option should be turned on whenever it is possible, as it has an impact on performance.

Camera
Use back face culling to hide the back side of the face.

Shadow
Use back face culling when casting shadows.

Light Probe Volume
Use back face culling when baking Light Probe Volumes. Additionally helps rejecting capture point inside the object to avoid light leaking

Displacement
Controls how the displacement output from the shader node tree is used.

Bump Only::	Use Bump Mapping to simulated the appearance of displacement. This only modifies the shading normal of the object. Vertex position is not affected.
Displacement Only::	This mode is not supported and falls back to *Displacement and Bump*.
Displacement and Bump::	Combination of true displacement and bump mapping for finer details. Vertex position is modified.

Note

This type of displacement is not precomputed. It has a performance impact multiplied by the render sample count. However, the evaluation is much faster than doing it using geometry nodes or a displacement modifier.

Note

Displacing flat shaded geometry will split adjacent faces. This can be worked around by passing the vertex normals as a custom attribute.

Max Distance

The maximum distance a vertex can be displaced when using true displacement. Displacements over this threshold may cause visibility issues. These visibility issues can be observed when the object is out of view at the edge of screen with parts being displaced inside the view. The object would then disappear because of camera culling. This can also produce missing shadow updates where the displaced geometry is.

Transparent shadows

Use transparent shadows for this material if it contains a Transparent BSDF. Disabling will render faster but not give accurate shadows.

Render Method

Controls the blending and the compatibility with certain features.

Dithered::	Allows for grayscale hashed transparency, and compatible with render passes and raytracing. Also know as deferred rendering. When using *Dithered* render method, the materials are rendered in layers. Each layer can only transmit (e.g. refract) light emitted from previous layers. If no intersection with the layers below exists, the transmissive BSDFs will fallback to light probes. Raytraced Transmission Use raytracing to determine transmitted color instead of using only light probes. This prevents the surface from contributing to the lighting of surfaces not using this setting.
Blended::	Allows the colored transparency, but incompatible with render passes and raytracing. Also known as forward rendering.

Sorting Problem

When using *Blended* render method, the order in which the color blending happens is important as it can change the final output color. EEVEE does not support per-fragment (pixel) sorting or per-triangle sorting. Only per-object sorting is available and is automatically done on all transparent surfaces based on object origin. Opaque surfaces (i.e. that have no transparency) will still have correct sorting regardless of the render method.

Tip

Face order can be adjusted in edit mode by using sort element or using a geometry node.

Note

Per-object sorting has a performance cost and having thousands of objects in a scene will greatly degrade performance.

Transparency Overlap

If enabled, all transparent fragments will be rendered. If disabled, only the front-most surface fragments will be rendered. This option can be disabled to fix sorting issues caused by blending order. Only available for the *Blended* render method.

Thickness

Determines what model to use to approximate the object geometry.

Sphere::	Approximate the object as a sphere whose diameter is equal to the thickness defined by the node tree. This is more suited to objects with rounder edges (e.g. a monkey head), and is perfectly suited to spheres.
Slab::	

Approximate the object as an infinite slab of thickness defined by the node tree. This is more suited to very flat or thin objects (e.g. glass panels, grass blades).

From Shadow

Use the shadow maps from shadow casting lights to refine the thickness defined by the material node tree. This takes the minimum thickness between the shadow map and the material node tree value. This is useful for objects where pre-computation is difficult (e.g. complex meshes), impossible (e.g. procedural geometry with displacement) or just impractical. However, this will have a performance impact that scale with the number of render samples.

Volume

Intersection

Determines which inner part of the mesh will produce volumetric effect.

Fast::	Each face is considered as a medium interface. Gives correct results for manifold geometry that contains no inner part.
Accurate::	Faces are considered as medium interface only when they have different consecutive facing. Gives correct results as long as the max ray depth is not exceeded. Has significant memory overhead compared to the fast method.

Light Settings

Reference

Panel:: *Properties* ▸ *Light Shader Editor* ▸ *Sidebar* ▸ *Options*

Besides lighting from the background and materials with emission shaders, lights are another way to add light into the scene. The difference is that they are not directly visible in the rendered image, and can be more easily managed as objects of their own type.

See Light settings for settings common to all renderers.

Shadow

EEVEE uses a technique called *Virtual Shadow Mapping* along with *Shadow Map Raytracing*. *Virtual Shadow Mapping* produces more accurate results than traditional shadow mapping by putting resolution only where it is needed. It also includes a very efficient caching mechanism. This technique offers better performance than ray tracing and is compatible with any Render Method.

Tip

- The error message *Shadow buffer full* means that the system cannot allocate enough shadow memory. Increasing the Shadow Pool Size or the Resolution Limit on some lights can fix the issue. Otherwise, the only workaround is to disable shadow casting on some lights.
- *Shadow Map Raytracing* can be tweaked in the Render Settings.
- Turning on Jitter can reduce the light leaking artifacts caused by large lights and *Shadow Map Raytracing*.

See also

Limitations.

Jitter

Enable jittered soft shadows to increase shadow precision. Has a high performance impact as the shadow map cannot be cached and needs to be updated for each render sample.

Note

The effect isn't visible by default in the viewport. See render settings.

Overblur

Apply shadow tracing to each jittered sample to reduce under-sampling artifacts.

Note

Any value higher than zero will result in a blurrier shadow and is not physically correct.

Filter

Blur shadow aliasing using PCF with a circular kernel. The effective world scale of the filter depends on the shadow map resolution at the shadowed pixel position.

Note

Any value bigger than 1px will increase the chances of light leaking artifacts.

Resolution Limit

Minimum size of a shadow map pixel. Higher values use less memory at the cost of shadow quality. Higher values also speed-up rendering of heavy scenes. Each shadow is scaled depending on the shadowed pixel on screen. This can create very sharp shadows but also requires a lot of memory if the shadowed pixel is close to the camera. This property limits the maximum amount of detail that the shadow map can capture.

Note

Reducing the shadow map resolution will increase the chances of light leaking artifacts.

Absolute Resolution Limit

Limit the resolution at 1 unit from the light origin instead of relative to the shadowed pixel. This makes Resolution Limit act as a regular shadow map pixel size.

Hint

With this option enabled, the following equation can be used to set the *Resolution Limit* with a desired resolution:

$$resolution\_limit = 2 * \sqrt{2} / resolution$$

The $2 * \sqrt{2}$ refers to the unit cube diagonal and $resolution$ refers to the desired resolution (e.g. 1024px).

Note

The setting Absolute Resolution Limit does not exist for Sun Light.

Influence

These parameters modulate the intensity of the light depending on the shader type. These are meant for artistic control, and any value other than 1.0 breaks PBR rules.

Diffuse
    Diffuse reflection intensity multiplier.

Glossy
    Glossy light intensity multiplier.

Transmission
    Transmission light intensity multiplier.

Volume Scatter
    Volume light intensity multiplier.

Custom Distance

If enabled, uses Distance as the custom attenuation distance instead of global Light Threshold. In order to avoid long setup times, this distance is first computed automatically based on a light threshold. The distance is computed at the light origin and using the inverse square falloff.

Distance
    Specifies where light influence will be set to 0.

Note

The setting Custom Distance does not exist for Sun Light.

See also

Global Light Threshold.

Light Probes

Light probe objects are used by EEVEE as support objects.

There are three different types of light probes. Each type of light probe records the lighting at a different resolution and frequency. Probes are used together to recover incoming light information when using ray tracing is not possible (either for performance or for technical limitations).

These types of objects are only useful for EEVEE (and by extension, the Material Preview mode).

Types

- Sphere
  - Properties
- Plane
  - Placement
  - Properties
- Volume
  - Properties

Light Probe Sphere

A light probe sphere records the light incomming from many directions at a single location.

They are used for smooth and semi-rough reflections. Sphere probes smoothly blend to light probe volume lighting for completely diffuse reflections.

If *Raytracing* is turned on, they are used as a fallback if a ray misses.

Note

In both usages, the light probe spheres are shadowed by light probe volume. This is done in order to reduce light leaking in shadowed areas and reduce the need to setup more light probe spheres.

Adjusting their resolution is done inside the Scene data panel.

The world also has an internal light probe sphere with a resolution that can be adjusted in the World data panel.

Properties

Reference

Panel::	*Object Data* ‣ *Probe*

Type
: Shape of the influence volume. Can be set to Sphere or Box.

Radius

A probe object only influences the lighting of nearby surfaces. This influence zone is defined by the size parameter and object scaling.

Falloff
: Percentage of the influence distance in which the influence of a probe fades linearly.

Capture

Note

In the viewport, capture only happens if an update is detected on the light probe data or position. For renders, the capture happens at the start of each frame.

Clipping
: Define the near and far clip distances when capturing the scene.

Custom Parallax

Reference

Panel::	*Object Data ‣ Custom Parallax*

By default, the influence volume is also the parallax volume. The parallax volume is a volume on which the recorded light is projected. It should roughly fit it surrounding area. In some cases it may be better to adjust the parallax volume without touching the influence parameters. In this case, enable the *Custom Parallax* and change the shape and radius of the parallax volume independently.

Viewport Display

Data
: Show the captured light using a reflective sphere of the given size.

Clipping

Show the clipping distance in the 3D Viewport.

Influence

Show the influence bounds in the 3D Viewport. The inner sphere is where the falloff starts.

Parallax

Show the *Custom Parallax* shape in the 3D Viewport.

Light Probe Plane

A light probe plane records the light incoming from a single direction for all visible points on a plane. The specular reflection direction is the only one currently available.

This type of light probe is suited to smooth planar surfaces.

Each visible planar light probe increases the render time as the scene needs to be rendered for each of them.

Light probe planes only work when the ray tracing method is set to `Screen-Trace`. When enabled, they accelerate the tracing process and complete the missing data from the screen space ray tracing.

Note

Reflections and volumetrics are not supported inside Light probe planes.

Placement

If Backface Culling is not enabled, snapping the light probe plane to the planar surface will effectively capture the underside of the surface.

You can manually move the light probe plane above the surface enough for it to not appear in the capture. Alternatively you can disable the light probe visibility in the object visibility panel.

Properties

Reference

Panel:: *Object Data ‣ Probe*

Distance

A probe object only influences the lighting of surfaces inside its influence zone. This influence zone is defined by the distance parameter and the object's scale.

For light probe planes, the influence distance is the distance from the plane. Only surfaces whose normals are aligned with the Reflection Plane will receive the captured reflection.

Capture

Clipping Offset

Define how far below the plane the near clip is when capturing the scene. Increasing this can fix reflection contact problems.

Viewport Display

Reference

Panel:: *Object Data ‣ Viewport Display*

Arrow Size

Size of the arrow showing the reflection plane normal.

Capture

Show the captured reflected image onto a fully reflective plane in the 3D Viewport.

Influence

Show the influence bounds in the 3D Viewport.

Light Probe Volume

A volume probe records the light incomming from all directions at many locations inside a volume.

The light is then filtered and only the diffuse light is recorded. The capture point positions are visible as an overlay when the Irradiance Volume object is selected.

If an object is not inside any Irradiance Volume, or if the indirect lighting has not been baked, the world's diffuse lighting will be used to shade it.

Tip

- When lighting indoor environments, try to align grids with the room shape.
- Try not to put too much resolution in empty areas or areas with a low amount of lighting variation.
- Bad samples can be fixed by adding a smaller grid near the problematic area.
- Large scenes may require using many volumes with different level of details.

Properties

Reference

Panel::	*Object Data ‣ Probe*

Intensity

Intensity factor of the recorded lighting. Making this parameter anything other than 1.0 is not physically correct. To be used for tweaking, animating or artistic purposes.

Sampling Bias

Normal Bias

Offset sampling of the irradiance grid in the surface normal direction to reduce light bleeding. Can lead to specular appearance of diffuse surface if set too high.

View Bias

Offset sampling of the irradiance grid in the viewing direction to reduce light bleeding. Can lead to view dependant result if set too high. Prefer this if camera is static.

Facing Bias

When set to zero, avoids capturing points behind the shaded surface to bleed light onto the shaded surface. This produces non-smooth interpolation when the capture resolution is high. Increasing this bias will make the interpolation smoother but also introduce some light bleeding.

Validity & Dilation

During the baking process, a validity score is assigned to each capture point. This score is based on the number of back-faces hit when capturing the incoming lighting. Only materials with *Single Sided* turned on for Light Probe Volumes will reduce the validity score.

Validity Threshold

Capture points with validity below this threshold will be ignored during lighting interpolation. This remove the influence of capture points trapped inside closed geometry, reducing the artifacts they produced.

Dilation Threshold

Capture points with validity below this threshold will have their data replaced using valid neighbors.

Radius

Radius in capture points in which to search for a valid neighbor.

Bake

Light probe volume light data is static and needs to be manually baked. Once baked, the data is stored inside the object data-block and can be moved, animated and linked between blender files.

Note

Baking uses the render visibility of the objects in the scene.

During baking, the scene is converted into a different representation to accelerate light transport. This representation can be very memory intensive and prevents baking if it cannot fit inside the GPU memory. There are a few way to deal with this issue:

- Larger scenes should be divided into smaller sections or use different level of details.
- Reduce *Surfel Resolution*.
- Turn off the light probe volume visibility option on objects that have little to no effect in the bake.

Tip

The internal scene representation can be inspected using the `Debug Value` 3, 4 and 5.

Resolution

Resolution X, Y, Z

Spatial resolution for volumetric light probes is determined per probe. The local volume is divided into a regular grid of the specified dimensions. The lighting will be captured for each cell in this grid.

Bake Samples

Number of ray directions to evaluate when baking. This increases the baking time proportionally to the size of the scene representation.

Surfel Resolution

Number of surfels to spawn in one local unit distance. Higher values increase quality, but have a huge impact on memory usage.

Tip

A good value is twice the maximum *Resolution*.

Capture

Capture Distance

Distance around the light probe volume that will be captured during the bake. A distance of 0 will only considered the inside of the volume.

World Contribution

Bake incoming light from the world instead of just visibility for more accurate lighting, but lose correct blending to surrounding irradiance volumes.

Indirect Light Contribution

Capture light bounces from light source.

Emission Contribution

Capture emissive surfaces when baking.

Clamping

Direct Light

Clamp incoming direct light. 0.0 disables direct light clamping. Here direct light refers to the light that bounces only once (from the light object) or light coming from emissive materials.

Indirect Light

Clamp incoming indirect light. 0.0 disables indirect light clamping. Here indirect light refers to the light that bounces off a surface after the first bounce (from the light object) or during the first bounce if the light comes from emissive materials.

Tip

Setting *Clamp Indirect* to a very non-zero value will effectively only record the first light bounce leading.

Offset

In order to reduce artifacts caused by bad capture point positioning, the bake process adjusts their location before capturing light. It moves the capture points slightly away from surrounding surfaces and tries to move them out of objects if they are not too far bellow the surface.

Surface Offset
  Distance to move the capture points away from surfaces.

Search Distance
  Distance to search for valid capture positions if the capture point is near the back-face of a single-sided object.

Note

Only materials with *Single Sided* turned on for Light Probe Volumes will move capture point position.

Viewport Display

Data
  Show the captured light using small diffuse spheres of the given size.

Influence

Show the influence bounds in the 3D Viewport. The inner sphere is where the falloff starts.

Clipping
: Show the clipping distance in the 3D Viewport.

Limitations

- Limitations
    - Attributes and Properties
    - Cameras
    - Lights
    - Light Probes
    - Indirect Lighting
    - Shadows
    - Volumetrics
    - Depth of Field
    - Screen Space Effects
    - Shader Nodes
    - Memory Management
    - CPU Rendering
    - Multiple GPU Support
    - Headless Rendering
- Supported Nodes
    - EEVEE only Nodes
    - Other Nodes Support

Limitations

EEVEE's goal is to be an interactive render engine. Some features may not be there yet or may be impossible to implement into EEVEE's architecture without compromising performance.

Here is a rather exhaustive list of all the limitations you can expect while working with EEVEE.

Attributes and Properties

- Only 14 attributes from Geometry Nodes are supported in a material
- Only 8 custom object properties are supported in a material

Cameras

- Only perspective and orthographic projections are currently supported.

Lights

- Lights can only have one color and do not support light node trees.
- Unlike in Cycles, the Size of spot lights does not change the softness of the cone.
- The area light Beam spread option is not supported.

Light Probes

- EEVEE supports up to 128 active light probe spheres.
- EEVEE supports up to 16 active light probe planes inside the view frustum.
- Active light probe volumes must fit inside the Light Probes Volume Memory Pool.

Indirect Lighting

- Light probe capture does not support specular reflections. Specular energy is treated as diffuse.

Shadows

- *Shadow Map Raytracing* can produce light leaking because of overlapping shadow casters. This can be mitigated by using lower step count, enabling jitter, or reducing the light shape size.
- Thin objects (e.g. walls without thickness) might have light leaking on the shadowed side. This can be mitigated by making the object have some thickness or lowering Resolution Limit.

Volumetrics

- Only single scattering is supported.
- Volumetrics are rendered only for the camera “rays”. They don’t appear in reflections/refractions and probes.
- Volumetric shadowing only work in volumetrics. They won’t cast shadows onto solid objects in the scene.
- Volumetric shadowing only work for volumes inside the view frustum.

Depth of Field

- Blended materials cannot be correctly handled by the post-processing blur, but will be correctly handled by the sample-based method. For this, you need to disable the post-process depth of field by setting the *Max Size* to 0.

Screen Space Effects

Ray-triangle intersection is not currently supported. Instead of this, EEVEE uses the depth buffer as an approximated scene representation. This reduces the complexity of scene scale effects and enables a higher performance. However, only what is in inside the view can be considered when computing these effects. Also, since it only uses one layer of depth, only the front-most pixel distance is known.

These limitations creates a few problems:

- The screen space effects disappear when reaching the screen border. This can be partially fixed by using the *overscan* feature.
- Screen space effects lack deep information (or the thickness of objects). This is why most effects have a thickness parameter to control how to consider potential intersected pixels.
- Objects behind other objects (occluded) are not considered by these effects.
- Blended surfaces are not considered by these effects. They are not part of the depth prepass and do not appear in the depth buffer.
- Objects that a part of Holdout Collections will not be rendered with screen space effects.

Raytracing

- Blended materials and materials using raytrace refractions will not appear in dithered materials reflections.
- Blended materials are not compatible with raytracing.
- Only one refraction event is correctly modeled. An approximation of the second refraction event can be achieved using the Thickness workflow.
- Only dithered materials *not* using Raytrace Refractions can be refracted.

Shader Nodes

- All BSDF’s are using approximations to achieve realtime performance so there will always be small differences between Cycles and EEVEE.
- Some utility nodes are not yet compatible with EEVEE.
- Certain combinations of BSDF’s will result in more noise than others. This is the case when mixing Diffuse BSDF and Refraction BSDF.

- Displacement of flat shaded surfaces will split the mesh into triangles. See Displacement for a workaround.

See also

For a full list of unsupported nodes see Nodes Support.

Memory Management

In EEVEE, GPU Memory management is done by the GPU driver. In theory, only the needed textures and meshes (now referred as “the resources”) for a single draw call (i.e. one object) needs to fit into the GPU memory.

So if the scene is really heavy, the driver will swap things in and out to make sure all objects are rendered correctly.

In practice, using too much GPU memory can make the GPU driver crash, freeze, or kill the application. So be careful of what you ask.

There is no standard way of estimating if the resources will fit into the GPU memory and/or if the GPU will render them successfully.

CPU Rendering

Being a rasterization engine, EEVEE only uses the power of the GPU to render. There is no plan to support CPU (software) rendering as it would be very inefficient. CPU power is still needed to handle high scene complexity as the geometry must be prepared by the CPU before rendering each frame.

Multiple GPU Support

There is currently no support for multiple GPU systems.

Headless Rendering

Headless rendering is not supported on headless Windows systems.

Supported Nodes

Most nodes are taken from Cycles. However, some features are missing and may (or may not) be implemented in EEVEE in the future.

See also

Shader Nodes.

EEVEE only Nodes

These nodes are only available if EEVEE is the active render engine. These nodes will not work in Cycles.

Shader to RGB

EEVEE supports the conversion of BSDF outputs into color inputs to make a wide variety of custom shading. This is supported using the Shader to RGB node. This node evaluates the lighting of the BSDFs connected to it just like a *Blended* material and inherits its limitation.

Specular BSDF

This node implements the specular workflow found in other render engines.

Other Nodes Support

If something is not listed here, it is supported.

Shader Nodes

In the general case, shader nodes should behave more or less like in Cycles. So be sure to check out the Cycles section of this manual for that.

See also

Materials.

Although most BSDFs are supported, many of them are approximations and are not feature complete.

Diffuse BSDF
: Roughness is not supported. Only Lambertian diffusion is supported.

Glass / Refraction BSDF
: Only supports GGX and Multiscatter GGX distribution. See Raytracing limitations.

Glossy BSDF
: Only supports GGX and Multiscatter GGX distributions.

Subsurface Scattering
: Random Walk sampling, IOR and Anisotropic are not supported.

Transparent BSDF
: Colored and additive transparency are only compatible with blended modes.

Translucent BSDF
: Does not diffuse the light inside the object. It only lights the object with reversed normals.

Principled BSDF
: Cumulative limitations from Diffuse BSDF, Glossy BSDF, Refraction BSDF and Subsurface Scattering. Anisotropy is not supported. The Sheen layer is a crude approximation.

Volume Absorption

See Volume Limitation.

Volume Scatter
The anisotropy parameter will be mixed and averaged for all overlapping volumetric objects, which is not physically correct and differs from Cycles. Also see Volume Limitation.

Principled Volume
Same as Volume Scatter. See Volume Limitation.

Holdout
Partially supported, using dithered mode may give incorrect results.

Anisotropic BSDF
Not supported.

Toon BSDF
Not supported.

Hair BSDF
Not supported.

Sheen BSDF
Not supported.

Principled Hair BSDF
Not supported.

Input Nodes

Ambient Occlusion
The *Only Local* option is not supported.

Geometry
Pointiness is not supported.

Random per Island
Random per Island is not supported.

Attribute

Defaults to active UV layer. Only “density”, “color”, “flame” and “temperature” built-in Geometry attributes are supported. UVs and Color Attributes are supported. Only up to 8 Object or Instancer attributes per material (both types share the same limit), and 512 View Layer attributes per scene are supported.

Bevel

Not supported.

Curves Info

The Random output uses a different RNG algorithm. Range and statistical distribution of the values should be the same but the values will be different.

Light Path

EEVEE has no real concept of rays. But in order to ease the workflow between Cycles and EEVEE some of the outputs are only supported in particular cases. This node makes it possible to tweak indirect lighting in the shader.

- *Is Camera*: Supported.
- *Is Shadow*: Supported.
- *Is Diffuse*: Set to 1.0 when baking light probe volume. Otherwise is set to 0.0.
- *Is Glossy*: Set to 1.0 when baking light probe sphere or plane. Otherwise is set to 0.0.
- *Is Singular*: Not supported. Same as Is Glossy.
- *Is Reflection*: Not supported. Same as Is Glossy.
- *Is Transmission*: Not supported. Same as Is Glossy.
- *Ray Length*: Not supported. Defaults to 1.0.
- *Ray Depth*: Not supported. Defaults to 0.0.
- *Diffuse Depth*: Partially supported. Set to 1.0 when baking light probe volume. Otherwise is set to 0.0.
- *Glossy Depth*: Partially supported. Set to 1.0 when baking light probe sphere or plane. Otherwise is set to 0.0.
- *Transparent Depth*: Not supported. Defaults to 0.

- *Transmission Depth*: Not supported. Same as Glossy Depth.

Note

Is Glossy does not work with Screen Space Reflections/Refractions but does work with reflection planes (whether used with SSR or not).

Particle Info
Not supported.

Texture Coordinate
From Instancer is not supported.

UV Map
From Instancer is not supported.

Wireframe
Pixel size option does not give exactly the same output as Cycles. The width can be a bit different.

Texture Nodes

Most texture nodes are supported except for the exceptions listed below:

IES Texture
Not supported.

Image Texture
Smart Interpolation always uses Cubic interpolation. Artifact present using Tube or Sphere projection with linear interpolation. This is due to hardware mip-mapping and Anisotropic filtering. This kind of artifact will be also visible if the texture coordinates provided are not continuous. Using Box projection with *Extend type* set to Clip or Extend is not supported. Instead, it will always use Repeat.

Point Density
Not supported.

Sky Texture
    In Nishita mode, the *Sun Disc* property is not supported.

Other Nodes

Light Falloff
    Not supported.

Cycles

- Introduction
- Render Settings
    - Grease Pencil
    - Sampling
    - Light Paths
    - Volumes
    - Subdivision
    - Curves
    - Simplify
    - Motion Blur
    - Film
    - Performance
- World Settings
- Object Settings
    - Object
    - Adaptive Subdivision
    - Cameras
- Material Settings
- Light Settings
- GPU Rendering
- Experimental Features
- Render Baking
- Optimizing Renders
    - Reducing Noise
    - Shader Nodes

Introduction

57
BARBERSHOP
DE TIJGER

Cycles is Blender's physically-based path tracer for production rendering. It is designed to provide physically based results out-of-the-box, with artistic control and flexible shading nodes for production needs.

To use Cycles, select it as the *Render Engine* in the Render properties. For GPU accelerated rendering, enable compatible devices in *Preferences* ▸ *System* ▸ *Cycles Render Devices*.

See also

The Cycles website with more information and a gallery.

Render Settings

- Grease Pencil
- Sampling
  - Adaptive Sampling
  - Denoising
  - Path Guiding
  - Lights
  - Advanced
- Light Paths
  - Ray Types
  - Bounce Control
  - Transparency
  - Settings
- Volumes
- Subdivision
- Curves
  - Viewport Display
- Simplify
  - Viewport
  - Render
  - Culling
  - Grease Pencil
- Motion Blur
  - Shutter Curve
  - Limitations
- Film
  - Pixel Filter
  - Transparent
- Performance
  - Threads
  - Memory
  - Acceleration Structure
  - Final Render
  - Viewport

Grease Pencil

Reference

Panel:: *Render ‣ Grease Pencil*

This panel is comprised of settings to control the rendering of Grease Pencil Lines.

Anti-Aliasing Threshold

Threshold for the edge detection algorithm used to correct aliasing, higher values might over blur some part of the image.

Sampling

Reference

Panel:: *Render ‣ Sampling*

The integrator is the rendering algorithm used to compute the lighting. Cycles currently supports a path tracing integrator with direct light sampling. It works well for various lighting setups, but is not as suitable for caustics and some other complex lighting situations.

Rays are traced from the camera into the scene, bouncing around until they find a light source such as a light, an object emitting light, or the world background. To find lights and surfaces emitting light, both indirect light sampling (letting the ray follow the surface BSDF) and direct light sampling (picking a light source and tracing a ray towards it) are used.

Viewport Samples
: Number of samples for viewport rendering. Setting this value to zero enables indefinite sampling of the viewport.

Render Samples
: Number of paths to trace for each pixel in the final render. As more samples are taken, the solution becomes less noisy and more accurate.

Time Limit
: Renders scene until time limit or sample count is reached. When the time is set to 0, the sample count is used to determine when the render stops.

> **Note**
>
> The time limit does not include pre-render processing time, only render time.

Adaptive Sampling

With adaptive sampling Cycles automatically reduces the number of samples in areas that have little noise, for faster rendering and more even noise distribution. For example hair on a character may need many samples, but the background may need very few.

With adaptive sampling it is also possible to render images with a target amount of noise. This is done by settings the *Noise Threshold*, typical values are in the range from 0.1 to 0.001. Then render samples can then be set to a high value, and the renderer will automatically choose the appropriate amount of samples.

Noise Threshold
: The error threshold to decide whether to continue sampling a pixel or not. Typical values are in the range from 0.1 to 0.001, with lower values meaning less noise. Setting it to exactly 0 lets Cycles guess an automatic value for it based on the total sample count.

Min Samples
: The minimum number of samples a pixel receives before adaptive sampling is applied. When set to 0 (default), it is automatically set to a value determined by the *Noise Threshold*.

Denoising

Denoising removes noise while previewing scenes in *Rendered* mode in the 3D Viewport or for final renders.

Render
: Denoising for the final render can be enabled or disabled with the checkbox. For denoising the image after rendering with the Denoising node, the Data Render Passes also adapt to the selected denoiser.

OpenImageDenoise::

	Uses Intel’s Open Image Denoise, an AI denoiser. Typically provides the highest quality, and is the default.
OptiX::	Uses NVIDIA’s OptiX AI denoiser. Supports GPU acceleration on some older NVIDIA GPUs where OpenImageDenoise does not. Only available on NVIDIA GPUs when configured in the Cycles Render Device user preferences.

Viewport

Denoising for the *Rendered* mode in the 3D Viewport can be enabled or disabled for with the checkbox.

Automatic::	Uses GPU accelerated denoising if supported, for best performance. Prefers OpenImageDenoise over OptiX.
OpenImageDenoise::	Uses Intel’s Open Image Denoise, an AI denoiser. Typically provides the highest quality.
OptiX::	Uses NVIDIA’s OptiX AI denoiser. Supports GPU acceleration on some older NVIDIA GPUs where OpenImageDenoise does not. Only available on NVIDIA GPUs when configured in the Cycles Render Device user preferences.

Passes

Controls which Render Pass the denoiser should use as input, which can have different effects on the denoised image. Generally, the more passes the denoiser has to denoise the better the result. It is recommended to at least use *Albedo* as *None* can blur out details, especially at lower sample counts.

None::	Denoises the image using color data.
Albedo::	Denoises the image using color and albedo data.

Albedo + Normal::	Denoises the image using color, albedo, and normal pass data.

Prefilter *OpenImageDenoise*

Controls whether or not prefiltering is applied to *Input Passes* for use when denoising. Visible only when using *OpenImageDenoise*.

None::	Does not apply any prefiltering to the input passes. This option retains the most detail and is the fastest, but assumes the input passes are noise free which may require a high sample count. If the input passes aren't noise free, then noise will remain in the image after denoising.
Fast::	Assumes the input passes are not noise free, yet does not apply prefiltering to the input passes. This option is faster than *Accurate* but produces a blurrier result.
Accurate::	Prefilters the input passes before denoising to reduce noise. This option usually produces more detailed results than *Fast* with increased processing time.

Quality *OpenImageDenoise*

Overall denoising quality. Visible only when using *OpenImageDenoise*.

High::	Produces the highest quality output at the cost of time.
Balanced::	Balanced between performance and quality.
Fast::	Produces an output fast at the cost of quality (ideal for viewport rendering).

Start Sample

Sample to start denoising in the 3D Viewport.

Use GPU

Perform denoising on the GPU. This is significantly faster than on CPU, but requires additional GPU memory. When large scenes need more GPU memory, this option can be disabled.

See GPU Rendering for details on supported GPU.

Path Guiding

Path guiding helps reduce noise in scenes where finding a path to light is difficult for regular path tracing, for example when a room is lit through a small door opening. Important light directions are learned over time, improving as more samples are taken. Guiding is supported for surfaces with diffuse BSDFs and volumes with isotropic and anisotropic scattering.

Note

- Path guiding is only available when rendering on a CPU.
- While path guiding helps render caustics in some scenes, it is not designed for complex caustics as they are harder to learn and guide.

Training Samples

The maximum number of samples to use for training. A value of 0 will keep training until the end of the render. Usually 128 to 256 training samples is enough for accurate guiding. Higher values can lead to a minor increases in guiding quality but with increased render times.

Surface

Enable path guiding for the diffuse and glossy components of surfaces.

Volume

Enable path guiding inside volumes.

Lights

Light Tree

Use a light tree to more effectively sample lights in the scene, taking into account distance and estimated intensity. This can significantly reduce noise, at the cost of a somewhat longer render time per sample.

Certain lighting properties are not accounted for in the light tree. This include custom falloff, ray visibility, and complex shader node setups including textures. This can result in an increase in noise in some scenes that make use of these features.

Note, this feature is currently disabled for AMD GPUs on macOS.

Light Threshold

Probabilistically terminates light samples when the light contribution is below this threshold (more noise but faster rendering). Zero disables the test and never ignores lights. This is useful because in large scenes with many light sources, some lights might only contribute a small amount to the final image, and increase render times. Using this setting can decrease the render times needed to calculate the rays which in the end have very little effect on the image.

Advanced

Pattern

The random sampling pattern used by the integrator.

Automatic::	Uses *Blue-Noise* (see below), but for viewport rendering, it optimizes for first sample quality for an interactive preview.
Classic::	Use pre-computed tables of Owen-scrambled Sobol for random sampling.
Blue-Noise::	Use a blue-noise pattern, which optimizes the frequency distribution of noise, for random sampling. This results in an output that appears smoother despite not being less noisy overall.

Seed

Seed value for integrator to get different noise patterns.

Use Animated Seed (clock icon)

Changes the seed for each frame. It is a good idea to enable this when rendering animations because a varying noise pattern is less noticeable.

Sample Offset

The number of samples to skip when starting render. This can be used to distribute a render across multiple computers then combine the images with `bpy.ops.cycles.merge_images`

Scrambling Distance

These properties are not compatible with *Blue-Noise* sampling patterns.

Automatic

Uses a formula to adapt the scrambling distance strength based on the sample count.

Viewport

Uses the *Scrambling Distance* value for the viewport rendering. This will make the rendering faster but may cause flickering.

Multiplier

Lower values Reduce randomization between pixels to improve GPU rendering performance, at the cost of possible rendering artifacts if set too low.

Min Light Bounces

Minimum number of light bounces for each path, after which the integrator uses Russian Roulette to terminate paths that contribute less to the image. Setting this higher gives less noise, but may also increase render time considerably. For a low number of bounces, it is strongly recommended to set this equal to the maximum number of bounces.

Min Transparent Bounces

Minimum number of transparent bounces (more specifically “passthroughs”). Setting this higher reduces noise in the first bounces, but can also be less efficient for more complex geometry like hair and volumes.

Layer Samples

When render layers have per layer number of samples set, this option specifies how to use them.

Use::	The render layer samples will override the set scene samples.
Bounded::	Bound render layer samples by scene samples.
Ignore::	Ignore render layer sample settings.

Light Paths

Reference

Panel:: *Render ‣ Light Paths*

Ray Types

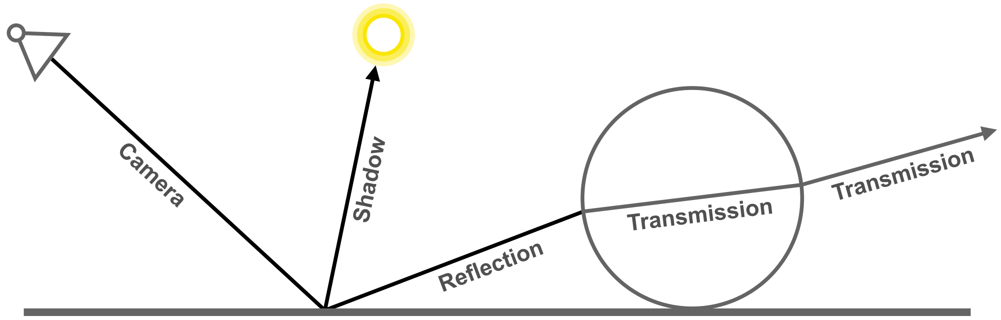
Ray types can be divided into four categories:

1. Camera: the ray comes straight from the camera.
2. Reflection: the ray is generated by a reflection off a surface.
3. Transmission: the ray is generated by a transmission through a surface.
4. Shadow: the ray is used for (transparent) shadows.

Reflection and transmission rays can further have these properties:

- Diffuse: the ray is generated by a diffuse reflection or transmission (translucency).
- Glossy: the ray is generated by a glossy specular reflection or transmission.
- Singular: the ray is generated by a perfectly sharp reflection or transmission.

The Light Path node can be used to find out the type of ray the shading is being computed for.



See also

The object ray visibility settings.

Bounce Control

The maximum number of light bounces can be controlled manually. While ideally this should be infinite, in practice a smaller number of bounces may be sufficient, or some light interactions may be intentionally left out for faster convergence. The number of diffuse reflection, glossy reflection and transmission bounces can also be controlled individually.

Light paths are terminated probabilistically when specifying a minimum number of light bounces lower than the maximum. In that case, paths longer than minimum will be randomly stopped when they are expected to contribute less light to the image. This will still converge to the same image, but renders faster while possibly being noisier.

Transparency

The Transparent BSDF shader is given special treatment. Rays pass straight through it, changing neither direction nor type as if there were no geometry at all.

Alpha pass output is also different for the transparent BSDF. Other transmission BSDFs are considered opaque, because they change the light direction. As such they cannot be used for alpha-over compositing, while this is possible with the transparent BSDF.

Note that, while semantically the ray passes through as if no geometry was hit, rendering performance is affected as each transparency step requires executing the shader and tracing a ray.

Settings

Max Bounces

Total

Maximum number of light bounces. For best quality, this should be set to the maximum. However, in practice, it may be good to set it to lower values for faster rendering. A value of 0 bounces results in direct lighting only.

Diffuse

Maximum number of diffuse bounces.

Glossy

Maximum number of glossy bounces.

Transmission

Maximum number of transmission bounces.

Volume

Maximum number of volume scattering bounces.

Transparent

Maximum number of transparency bounces.

Note, the maximum number of transparent bounces is controlled separately from other bounces. It is also possible to use probabilistic termination of transparent bounces, which might help rendering many layers of transparency.

Clamping

Direct Light

This option limits the maximum intensity a sample from rays which have not yet bounced can contribute to a pixel. It reduces noise at the cost of accuracy. Setting this option to 0.0 disables clamping altogether. Lower have a greater affect (dimmer samples) on the resulting image than higher values.

> **Note**
>
> This option provides a way to limit Fireflies. However, note that as you clamp out such values, other bright lights/reflections will be dimmed as well.
>
> Care must be taken when using this setting to find a balance between mitigating fireflies and losing intentionally bright parts. It is often useful to clamp indirect bounces separately, as they tend to cause more fireflies than direct bounces. See the *Clamp Indirect* setting.

Indirect Light

The same as *Direct Light*, but for rays which have bounced multiple times.

Caustics

A common source of noise is Caustics.

> **See also**
>
> See Reducing Noise for examples of the clamp settings in use.

Filter Glossy

When using a value higher than 0.0, this will blur glossy reflections after blurry bounces, to reduce noise at the cost of accuracy. 1.0 is a good starting value to tweak.

Some light paths have a low probability of being found while contributing much light to the pixel. As a result these light paths will be found in some pixels and not in others, causing Fireflies. An example of such a difficult path might be a small light that is causing a small specular highlight on a sharp glossy material, which is observed through a rough glossy material. In fact in such a case there practically occurs a caustic.

With path tracing it is difficult to find the specular highlight, but if you increase the roughness on the material, the highlight gets bigger and softer, and so easier to find. Often this blurring will hardly be noticeable, because it is blurred by the material anyway, but there are also cases where this will lead to a loss of detail in lighting.

Caustics

Reflective

While in principle path tracing supports rendering of caustics with a sufficient number of samples, in practice it may be inefficient to the point that there is just too much noise. This option can be unchecked, to disable reflective caustics.

Refractive

The same as above, but for refractive caustics.

Fast GI Approximation

Reference

Panel:: *Render ‣ Light Paths ‣ Fast GI Approximation*

Approximate diffuse indirect light with background tinted ambient occlusion. This provides fast alternative to full global illumination (GI), for interactive viewport rendering or final renders with reduced quality.

Method

Fast GI approximation method.

Replace::	Replace global illumination with ambient occlusion after a specified number of bounces.
Add::	Add ambient occlusion to diffuse surfaces.

AO Factor

The strength of the ambient occlusion.

AO Distance

Distance from shading point to trace rays. A shorter distance emphasizes nearby features, while longer distances make it also take objects farther away into account.

This option can also be overridden per object in the Object Properties, which is useful when you have both small and large scale objects in the same scene.

Viewport Bounces

Replace global illumination with ambient occlusion after the specified number of bounces when rendering in the 3D Viewport. This can reduce noise in interior scenes with little visual difference.

Render Bounces

Number of bounces when rendering final renders.

Volumes

Reference

Panel:: *Render ‣ Volumes*

Volume *Step* size is the distance between volume shader samples. Cycles automatically estimates this distance based on voxel size in volume objects and smoke simulations.

Render time can be reduced by increasing the step size, at the cost of potentially losing some volume detail. For procedural volume shaders that add detail, step size can be increased per object, material or world.

Step Rate Render
: Global multiplier on the step size for all volumes in renders. Increase to reduce render time, at the cost of losing detail.

Viewport
: Global multiplier on the step size for all volumes in the viewport. Increase for more responsive viewport rendering.

Max Steps
: Maximum number of steps through the volume before giving up, to protect from extremely long render times with big objects or small step sizes.

Subdivision

Reference

Panel:: *Render ▸ Subdivision*

Note

These settings are only available if Experimental Feature Set is turned on.

These settings are used to control Adaptive Subdivision.

Dicing Rate Render, Viewport
: Size of Micropolygons in pixels for the final/viewport render.

Offscreen Scale
: Multiplier for dicing rate of geometry outside of the camera view. The dicing rate of objects is gradually increased the further they are outside the camera view. Lower values provide higher quality reflections and shadows for off screen objects, while higher values use less memory.

Max Subdivisions
: Stop subdividing when this level is reached even if the dicing rate would produce finer Tessellation.

Dicing Camera
: Camera to use as reference point when subdividing geometry, useful to avoid crawling artifacts in animations when the scene camera is moving.

Curves

Reference

Panel:: *Render ‣ Curves*

These are global settings that apply to all instances of particle hair systems. The resolution of the strands is controlled by the step values in particle settings. Each hair system uses the material identified in the particle settings.

Shape

Rounded Ribbons::	Render curves as flat ribbon with rounded normals, for fast rendering. Curves are subdivided with a fixed number of specified subdivisions. Curve Subdivisions Number of subdivisions used in cardinal curve intersection (power of 2).
3D Curves::	Render curves as circular 3D geometry, for accurate results when viewing curves close up. Curves are automatically subdivided until the curve is smooth.

Viewport Display

These settings control the curve rendering settings used when the 3D viewport is set to Material Preview

Shape

Strand::	Render curves as a thin strand roughly a pixel wide. Curve diameter parameters are ignored with this setting.

Strip::	Render curves as a flat ribbon with rounded normals.

Additional Subdivisions

Additional subdivisions to be applied on top of the curve resolution set in the hair system settings. Increasing this value will smooth out the curves of the strands.

Simplify

Reference

Menu:: *Render ‣ Simplify*

Common Settings

Max Subdivision
: Maximum number of subdivision by the Subdivision Surface modifiers.

Child Particles
: Show only a subset of all child hairs and particles.

Texture Limit
: Automatically scales textures down so that they are no larger than the values chosen. This can help reduce computer memory resources when rendering large scenes with huge textures.

Viewport

See Common Settings above.

Volume Resolution
: Resolution percentage of volume objects in the viewport. This mostly affects memory usage rather than computation times.

Normals
: Skip computing custom normals and face corner normals for displaying meshes in the viewport.

Render

See Common Settings above.

Culling

Camera Cull
: Automatically culls objects based on the camera frustum defined by the *Margin*.

Distance Cull
: Automatically culls objects based on their distance from the active camera. This is set via the *Distance* property.

Grease Pencil

Playback Only
: Activates the simplification process only during animation playback.

Fill
: Shows the fill component in Grease Pencil materials.

Modifiers
: Shows Grease Pencil modifiers.

Shader Effects
: Shows Grease Pencil visual effects.

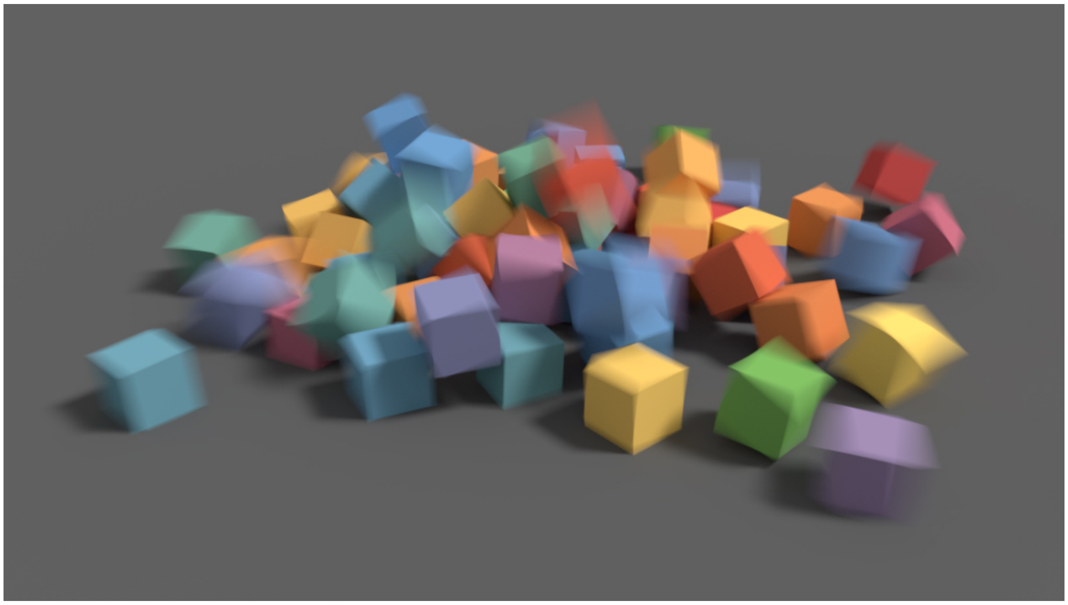
Layer Tinting
: Shows layers tint overrides.

Anti-Aliasing
: Use Anti-Aliasing to smooth stroke edges. The amount of anti-aliasing can be adjusted by the Anti-Aliasing Threshold.

Motion Blur

Reference	
Panel::	*Render ‣ Motion Blur*

Blender's animations are by default rendered as a sequence of *perfectly still* images. While great for stop-motion and time-lapses, this is unrealistic, since fast-moving objects do appear to be blurred in the direction of motion, both in a movie frame and in a photograph from a real-world camera.



Motion blur example. (blend-file)

Position
: Controls at what point the shutter opens in relation to the current frame.

Start on Frame::	Shutter is starting to open at the current frame.
Center on Frame::	Shutter is fully opened at the current frame.
End on Frame::	Shutter is fully closed at the current frame.

Shutter

Time (in frames) between when the shutter starts to open and fully closed. For example, shutter time 1.0 blurs over the length of 1 frame.

Rolling Shutter

Creates a Rolling Shutter effect.

None::	No rolling shutter effect.
Top-Bottom::	Renders rolling shutter from the top of the image to the bottom.

Rolling Shutter Duration

Controls balance between pure rolling shutter effect (if the value is zero) and pure motion blur effect (if the value is one).

Note

If there are particles or other physics system in a scene, be sure to bake them before rendering, otherwise you might not get correct or consistent motion blur.

See also

Each object has its own settings to control motion blur. These options can be found in the Object tab of the Properties. See object setting for more information.

Shutter Curve

Curve defining how the shutter opens and closes. The X axis is time, Y values of 0 mean fully closed shutter, Y values of 1 mean fully opened shutter. The default mapping is set to when shutter opens and closes instantly.

Limitations

- Camera motion blur does not work for Orthographic Cameras.
- Motion blur does not take into account the movement of Lights.

Film

Reference	
Panel::	*Render ▸ Film*

Exposure

This can be used to change the brightness of an image. Different than the *Exposure* option found in the Color management panel, this exposure option works *on the data* while the Color management exposure is on the *view transform*.

Pixel Filter

Due to limited resolution of images and computer screens, pixel filters are needed to avoid Aliasing. This is achieved by slightly blurring the image to soften edges.

Type

Pixel Filtering algorithm to use.

Box::	No filter.
Gaussian::	Smooth filter.
Blackman-Harris::	Default filter with a better balance between smoothness and detail preservation.

Width

Lower values give more crisp renders, higher values are softer and reduce aliasing.

Transparent

Render the background transparent, for compositing the image over another background after rendering.

Transparent Glass
: Render transmissive surfaces as transparent, for compositing glass over another background.

Roughness Threshold
: For transparent glass, keep surfaces with roughness above the threshold opaque.

Performance

Reference	
Panel::	*Render ‣ Performance*

Properties that affect render speeds or memory consumption. There are several presets available to help choose between different trade offs:

Default::	Balances memory saving and faster rendering settings.
Faster Render::	Uses settings to render faster at the cost of higher memory consumption.
Lower Memory::	Uses settings to decrease memory considered at the cost of slower renders.

Threads

Threads Mode

Method to determine the maximum number of CPU cores to use while rendering.

Auto-Detect::	Automatically chooses the amount of threads to match the number of logical processors on your computer.
Fixed::	Manually choose the maximum number threads to use for rendering. This can be useful for example, if you want to use your computer while rendering, you can set the property to a thread count lower the amount of logical processors on your computer.

Threads

The maximum number of CPU cores to use simultaneously while rendering.

Memory

Use Tiling

Render high resolution images in tiles to reduce memory usage. Tiles are cached to disk while rendering to save memory.

Tile Size

This value is used to control the size of the tile used for rendering; decreasing the size reduces memory usage.

> **Note**
>
> In some cases changing the *Tile Size* can result in increased performance. For example when a small object renders slowly compared to other objects, using a small *Tiles Size* can lead to an increase in performance.

Acceleration Structure

Use Spatial Splits

Spatial splits improve the rendering performance in scenes with a mix of large and small polygons. The downsides are longer BVH build times and slightly increased memory usage.

Use Curves BVH

Use a special type of BVH for rendering curves. The bounding boxes are not axis aligned allowing a spatially closer fit to the curve geometry. Disabling this option will reduce memory, at the cost of increasing curve render time.

BVH Time Steps

Split BVH primitives by this number of time steps to speed up render time at the expense of memory.

Use Compact BVH

Use a more compact BVH structure, which can reduce RAM usage but render slower.

Final Render

Persistent Data

Keep render data in memory after rendering for faster re-renders and animation renders at the cost of extra memory usage while performing other tasks in Blender.

When using multiple View Layers, only data from a single view layer is preserved to keep memory usage within bounds; however, objects shared between view layers are preserved.

Viewport

Pixel Size

Option to control the resolution for viewport rendering. Allows you to speed up viewport rendering, which is especially useful for displays with high DPI.

World Settings

Mist Pass

Reference

Panel::	*World* ▸ *Mist Pass*

Note

The mist pass must be enabled in the View Layer tab of the Properties Editor before the settings below are available in the World tab.

Mist can greatly enhance the illusion of depth in your rendering. To create mist, Blender generates a render layer with a depth map ranging between 0.0 and 1.0 that can be used in the Compositor to generate a mist effect.

Start
: The distance from the camera at which the mist starts to fade in.

Depth
: The distance from *Start* of the mist, that it fades in over. Objects further from the camera than *Start* + *Depth* are completely hidden by the mist.

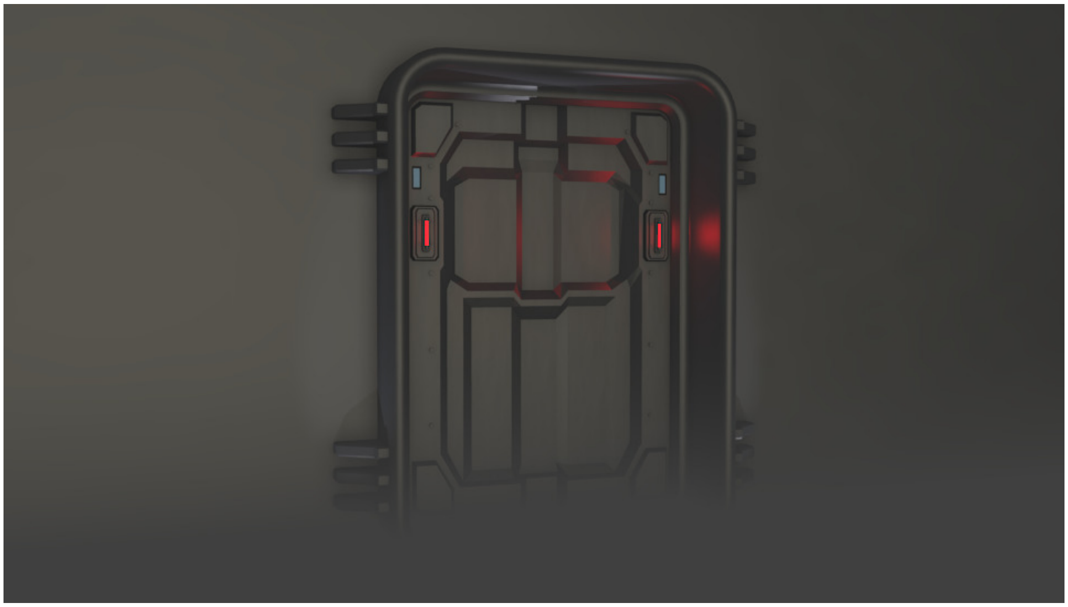
Falloff
: The curve function that controls the rate of change of the mist's strength further and further into the distance.

Quadratic::	Uses the same calculation as light falloff (\(1\over{x^2}\)) and provides the smoothest transition from transparent (0.0) to opaque (1.0).
Linear::	Has a steeper start than quadratic (\(1\over{x}\)).
Inverse Quadratic::	

Has the steepest start ($1\over{\sqrt{x}}$) and approaches 1.0 faster than the other two functions.

Tip

A visualization can be activated in the *Camera* ▸ *Viewport Display* panel.



Mist example (blend-file).

Ray Visibility

Reference

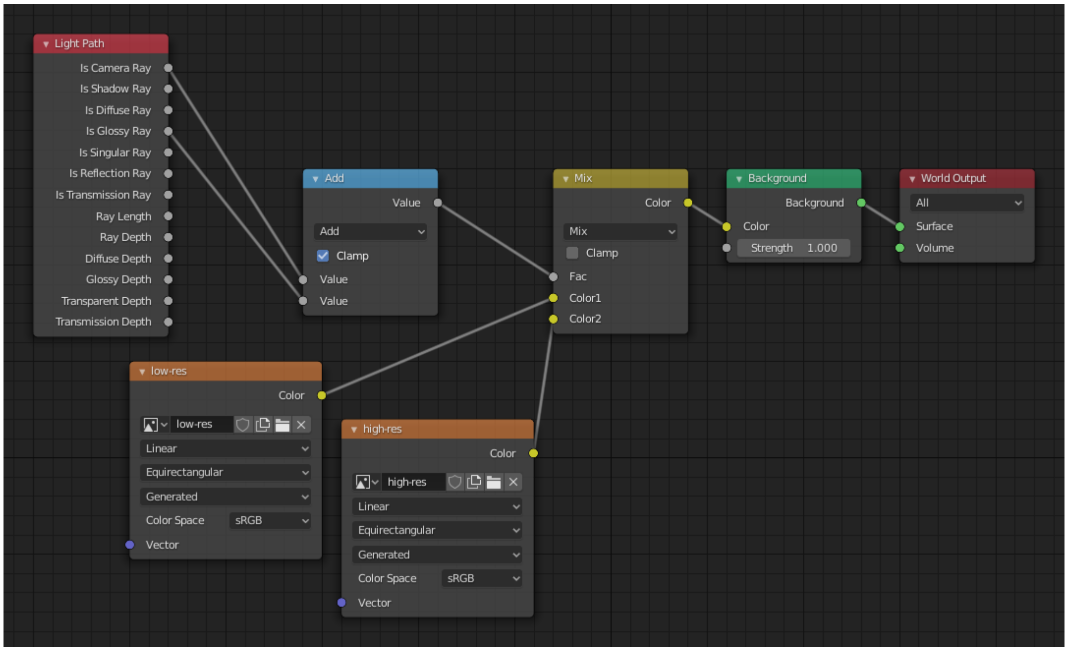
Panel:: *World* ▸ *Ray Visibility*

As with other objects, *Ray Visibility* allows you to control which other shaders can “see” the environment.

Tricks

Sometimes it may be useful to have a different background that is directly visible versus one that is indirectly lighting the objects. A simple solution to this is to add a Mix node, with the *Blend Factor* set to *Is Camera Ray*. The first input color is then the indirect color, and the second the directly visible color. This is useful when using a high-res image for the background and a low-res image for the actual lighting.

Similarly, adding the *Is Camera* and *Is Glossy* rays will mean that the high-res image will also be visible in reflections.


Light Path
Is Camera Ray
Is Shadow Ray
Is Diffuse Ray
Is Glossy Ray
Is Singular Ray
Is Reflection Ray
Is Transmission Ray
Ray Length
Ray Depth
Diffuse Depth
Glossy Depth
Transparent Depth
Transmission Depth
Add
Value
Add
Clamp
Value
Value
Mix
Color
Mix
Clamp
Fac
Color1
Color2
Background
Background
Color
Strength 1.000
World Output
All
Surface
Volume
low-res
Color
low-res
Linear
Equirectangular
Generated
Color Space sRGB
Vector
high-res
Color
high-res
Linear
Equirectangular
Generated
Color Space sRGB
Vector

Nodes for the trick above.

Settings

Reference

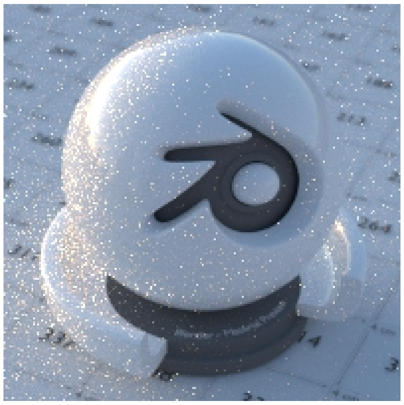
Panel:: *World ‣ Settings*

Surface

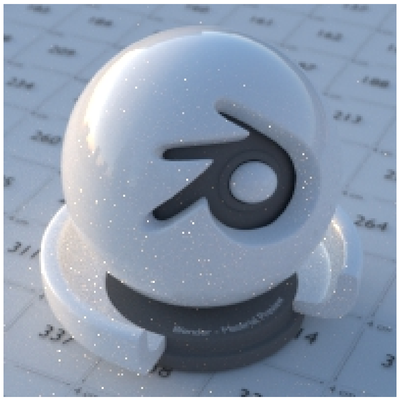
Sampling

Controls the sampling method for the world material. Selecting Auto or Manual enables *Multiple Importance Sampling* while None disables it. *Multiple Importance Sampling* is a method to sample the background texture such that lighter parts are favored, creating an importance map. It will produce less noise in the render in trade of artifacts (Fireflies). Enable this when using an image texture with small area lights (like the sun), otherwise noise can take a long time to converge.

Below is a comparison between *Multiple Importance Sample* off and on. Both images are rendered for 25 seconds (Off: 1,500 samples, On: 1,000 samples).



Multiple Importance Sample off.



Multiple Importance Sample on.

Map Resolution
: Sets the resolution of the importance map. A higher resolution will better detect small features in the map and give more accurate sampling but conversely will take up more memory and render slightly slower. Higher values also may produce less noise when using high-res images.

Max Bounces
: Maximal number of bounces the background light will contribute to the render.

See also

See Reducing Noise for more information on how to reduce noise.

Shadow Caustics

Mark the World Shader as a refractive caustic caster. This setting can be used in conjunction with the Cast and Receive caustics object settings to selectively speed up refractive caustic rendering of select objects.

Volume

Sampling Method

Distance::	For dense volumes lit from far away *Distance* sampling is more efficient in most cases. Usually this shouldn't be used for World volumes.
Equiangular::	If you have got a light inside or near the volume then *equiangular* sampling is better.
Multiple Importance::	If you have a combination of both, then the multiple importance sampling will be better.

Interpolation

Interpolation method to use for the volume.

Linear::	Simple interpolation which gives good results for thin volumes.
Cubic::	Smoothed high-quality interpolation needed for more dense volumes, but slower.

Homogeneous

Assume volume has the same density everywhere (not using any textures), for faster rendering. Usually this is automatically determined by the renderer. This settings provides a manual control for cases where it is not detected.

Step Size

Distance between volume shader samples for world volume shaders. See Volume Render Settings for more information.

Light Group

Light Group *Cycles only*

Select the Light Group to add the current *World Surface Shader* too.

Add Light Group

If the name input into the *Light Group* field does not align with an existing Light Group, then pressing this button will create a *Light Group* with that name and assign this *World Shader* to it.

Object Settings

Settings for objects and object data.

- Object
  - Visibility
  - Motion Blur
  - Shading
- Adaptive Subdivision
  - Known Limitations
- Cameras
  - Panoramic Cameras

Object

Visibility

Reference

Panel:: *Object Properties* ▸ *Visibility*

See also

There are several other general visibility properties.

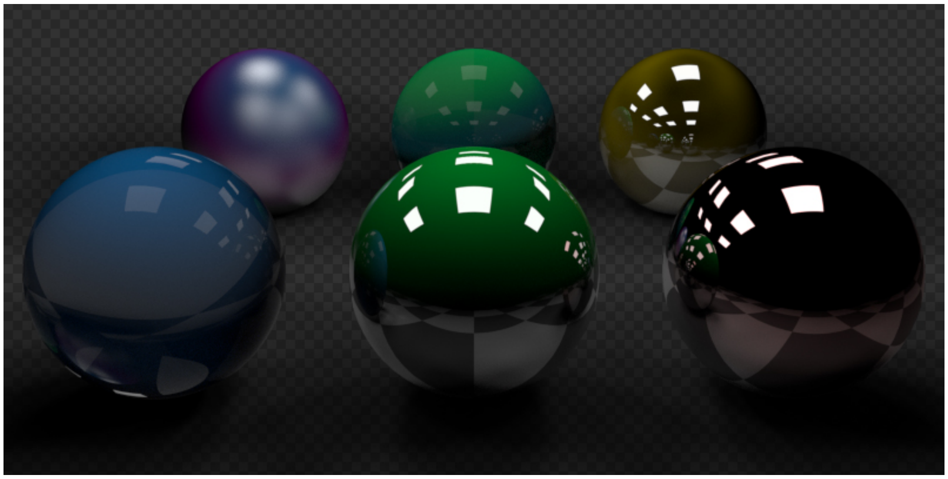
Mask

Shadow Catcher

Enables the object to only receive shadow rays. It is to be noted that, shadow catcher objects will interact with other CG objects via indirect light interaction. This simplifies compositing CGI elements into real-world footage.

Note

The *Shadow Catcher* outputs different results depending on if the *Shadow Catcher* pass is enabled in Render Layer settings. With the *Shadow Catcher* pass enabled, all indirect light interactions are captured. With it disabled, a simple approximation is used instead. The simple approximation is used in viewport rendering.



Example of the shadow catcher. Note how the material of the plane can still be viewed in the spheres.

Ray Visibility

Objects can be set to be invisible to particular ray types. This can be used, for example, to make an emitting mesh invisible to camera rays. For instanced objects, visibility is inherited; if the parent object is hidden for some ray types, the children will be hidden for these too.

In terms of performance, using these options is more efficient that using a shader node setup that achieves the same effect. Objects invisible to a certain ray will be skipped in ray traversal already, leading to fewer ray casts and shader executions.

Camera
: Makes the object visible to the Camera; this includes the viewport's perspective in viewport rendering.

Diffuse
: Makes the object visible in diffuse rays.

Glossy
: Makes the object visible in glossy rays.

Transmission
: Makes the object visible in transmission rays.

Volume Scatter
    Makes the object visible in volumetric scatter rays.

Shadow
    Enables the object to cast shadows.

Culling

In order to activate these options the respectively camera cull options have to be enabled in the scene simplify_panel.

Use Camera Cull
    Ignore and this way make objects invisible to rays outside of the camera frustum.

Use Distance Cull
    Will cull any objects further from the camera than a given distance. When used in combination with camera frustum culling, this can be used to avoid culling nearby objects that are outside the camera frustum, but still visible in reflections. It is also useful to cull small objects far from the camera.

Motion Blur

Reference

Panel::	*Properties ▸ Object Properties ▸ Motion Blur*

Each object has its own motion blur settings along with the Scene Level Motion Blur. These settings can be found in the Object Properties tab of the Properties.

Steps
    Controls accuracy of deformation motion blur, more steps uses more memory. The actual number of time steps is \(2^{steps -1}\).

Deformation

Enables motion blur for deformed meshes such as animated characters, including hair.

Warning

An object modifier setup that changes mesh topology over time can not render deformation motion blur correctly. Deformation blur should be disabled for such objects. Common examples of this are animated Booleans, Deformation before Edge Split, Remesh, Skin or Decimate modifiers.

Shading

Reference

Panel::	*Properties* ▸ *Object Properties* ▸ *Shading*

Shadow Terminator

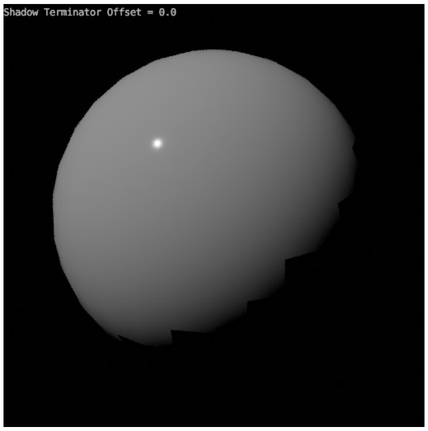
Geometry Offset

Offset rays from the surface to reduce shadow terminator artifacts on low-poly geometry. Higher values affect more triangles, a value of one affecting all triangles and zero having no affect. The default value only affects triangles at grazing angles to light and should eliminate most artifacts.

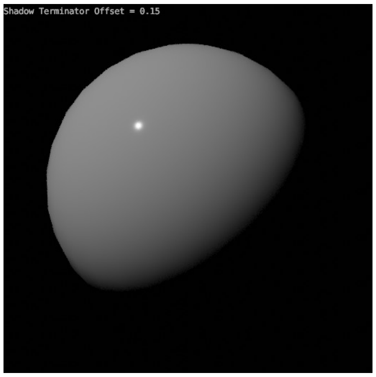
Unlike the *Shading Offset*, this option has little affect on the lighting making it the preferable method to handle shadow terminator artifacts.

Shading Offset

Pushes the shadow terminator (the line that divides the light and dark) towards the light to hide artifacts on low-poly geometry such as the ones below:



Shadow Terminator Artifacts.



Result of using an offset of 0.15.

Note

This property artificially alters the scene's lighting and is not energy conserving and consequently not physically accurate (see *Geometry Offset* instead).

Fast GI Approximation

AO Distance

Override for the world's AO Distance, if the value is zero the world's distance is used.

Caustics

Mark objects as caustic casters or receivers. This is used in conjunction with a Light or World Shader with *Shadow Caustics* enabled to selectively speed up caustic rendering of objects in your scene.

Note

The rendering technique used to speed up the rendering of caustics is based on MNEE. There are a number of limitations with this technique and it’s implementation in Cycles:

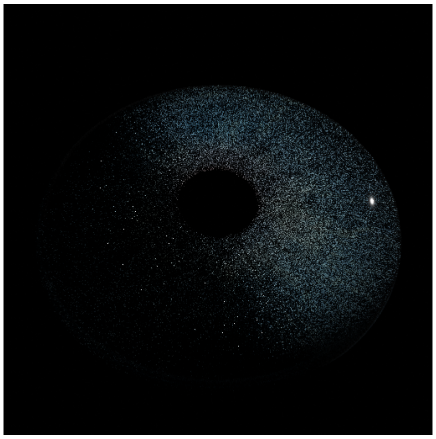
- Only refractive caustics in the shadows of objects work. Caustics from reflections or caustics that fall outside shadows are not rendered with this technique.
- MNEE Caustics are an approximation of caustics and will produce physically inaccurate results in many situations. Examples include incorrect brightnesses and the incorrect representation of caustics caused by rough or curved surfaces.
- In complex materials with multiple refractive BSDFs, MNEE will only generate caustics for one of the BSDFs.
- Filter Glossy settings are ignored when using MNEE for refractive caustics.
- MNEE Caustic rays can pass through up to 6 Caustic Caster surfaces between a Caustic Receiver and a Shadow Caustic light before the ray is terminated and caustics are ignored.
- The Ambient Occlusion and Bevel nodes will not produce a valid result on objects that are a Caustic caster or Caustic receiver while the scene contains a active Caustic caster, Caustic receiver, and Shadow Caustic Light.
- MNEE Caustics only work if the caustic caster has smooth normals.
- Volumetric materials are not considered when calculating MNEE caustics.
- Bump and normal maps are ignored when calculating caustics.

Cast Shadow Caustics

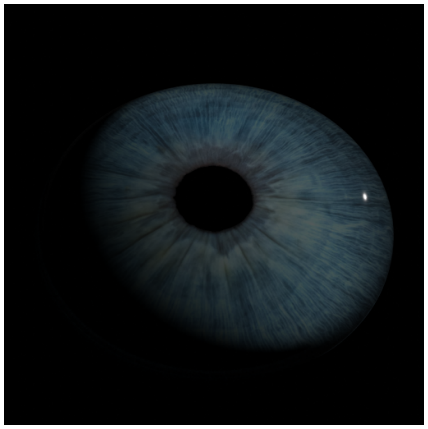
Mark an object as a caustic caster.

Receive Shadow Caustics

Mark an object as a caustic receiver.



Rendering caustics inside an eye without MNEE at 32 samples per pixel.



Rendering caustics inside an eye using MNEE at 32 samples per pixel.

Light Group

Light Group

Select the Light Group to add the current *Object* or *Light* too.

Add Light Group

If the name input into the *Light Group* field does not align with an existing Light Group, then pressing this button will create a *Light Group* with that name and assign the selected *Object* or *Light* to it.

Light Linking

Limit light influence to specified objects, with Light Linking.

Receiver Collection

Collection of objects that will receive light emitted from the object.

Shadow Linking

Limit shadows to specified objects, with Light Linking.

Shadow Blocker Collection

Collection of objects that will act as shadow blockers for light emitted from the object.

Adaptive Subdivision

Reference

Panel:: *Modifier ‣ Subdivision Surface*

Note

Implementation not finished yet, marked as an Experimental Feature Set.

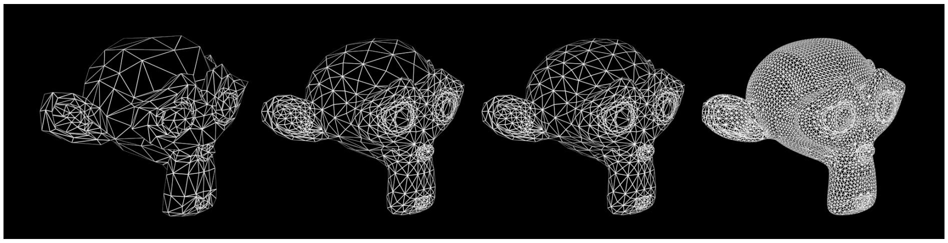
When using the *Experimental Feature Set* the Subdivision Surface Modifier gets changed to control the subdivision of a mesh at the time of rendering. For this, all the other settings are the same except the *View* and *Render* settings. These before mentioned settings get removed/renamed and the following settings are added:

Adaptive Subdivision

Use OpenSubdiv to give different subdivision levels to near and far objects automatically. This allows nearer objects to get more subdivisions and far objects to get less.

Dicing Scale

Multiplier of the scene dicing rate to determine the final size of Micropolygons in pixels.



Subdivision off/on, Dicing Scale: 1.0 - 0.3 - 0.05 (monkeys look identical in viewport, no modifiers).

Known Limitations

- Multi-user object data are currently made single users, leading to increased memory usage. For those it is better to use non-adaptive subdivision still.
- Multi-view renders can have some inconsistencies between views.

Warning

Instances are not tessellated individually. Instead, the original object is tessellated and then duplicated on all instances. To take advantage of both adaptive subdivision and instancing you should place the original object at the position of the instance that is closest from the camera.

Cameras

Panoramic Cameras

Cycles supports several types of panoramic cameras which are described in detail below. Note that these cannot be displayed in non-rendered modes in the viewport, i.e. *Solid* mode; they will only work for the final render.

Equirectangular

Render a panoramic view of the scenes from the camera location and use an equirectangular projection, always rendering the full 360° over the X axis and 180° over the Y axis.

This projection is compatible with the environment texture as used for world shaders, so it can be used to render an environment map. To match the default mapping, set the camera object rotation to (90, 0, -90) or pointing along the positive X axis. This corresponds to looking at the center of the image using the default environment texture mapping.

Latitude Min, Max
    Limits of the vertical field of view angles.

Longitude Min, Max
    Limits of the horizontal field of view angles.

Equiangular Cubemap Face

Improves on *Equirectangular* by providing a more uniform distribution of rendered pixel of the spherical environment. This results in an image that has little variation in visual resolution for the entire spherical projection. This is in contrast to *Equirectangular* which can lose detail in the poles of the image. This is also in contrast to cube map projections which lose detail near the edges of each face.

This panorama type is great for virtual reality use cases where providing as much visual detail for a limited resolution is important.

A limitation over *Equirectangular* is that this method does not have longitude or latitude limits.

Fisheye

Fisheye lenses are typically wide angle lenses with strong distortion, useful for creating panoramic images for e.g. dome projection, or as an artistic effect.

The *Fisheye Equisolid* lens will best match real cameras. It provides a lens focal length and field of view angle, and will also take the sensor dimensions into account.

The *Fisheye Equidistant* lens does not correspond to any real lens model; it will give a circular fisheye that does not take any sensor information into account but rather uses the whole sensor. This is a good lens for full-dome projections.

Lens
: Lens focal length in millimeters.

Field of View
: Field of view angle, going to 360 and more to capture the whole environment.

Fisheye Lens Polynomial

Match a real world camera by specifying the coordinates of a 4th degree polynomial.

The projection works as follows. Pixels in the image are mapped to positions (x, y) on the camera sensor in mm. A position on the sensor is mapped to a direction with spherical coordinates $(1, \theta, \phi)$ in radians as follows:

\[\begin{split}& r = \sqrt{x^2 + y^2}\\ & \theta = k_0 + k_1 r + k_2 r^2 + k_3 r^3 + k_4 r^4\\ & \phi = acos(x/r)\end{split}\]

Incoming light from this direction is then projected onto the corresponding pixel.

This can be used to model both fisheye and perspective cameras.

Mirror Ball

Render as if taking a photo of a reflective mirror ball. This can be useful in rare cases to compare with a similar photo taken to capture an environment.

Material Settings

Reference

Panel::	*Material* ▸ *Settings* and *Shader Editor* ▸ *Sidebar* ▸ *Settings*

Surface

Displacement

Method used to perform Displacement on materials.

Displacement Only::	Mesh vertices will be displaced before rendering, modifying the actual mesh. This gives the best quality results, if the mesh is finely subdivided. As a result, this method is also the most memory intensive.
Bump Only::	When executing the surface shader, a modified surface normal is used instead of the true normal. This is a less memory intensive alternative to actual displacement, but only an approximation. Surface silhouettes will not be accurate and there will be no self-shadowing of the displacement.
Displacement and Bump::	Both methods can be combined, to do displacement on a coarser mesh, and use bump mapping for the final detail.

Emission Sampling

The method used for sampling the emissive component of the material. This option will only have an influence if the material contains an emissive material node, otherwise it will be ignored.

None::	Do not use this surface as a light for sampling.
Auto::	Automatically determine if the surface should be treated as a light for sampling based on emission intensity.
Front::	Treat only the front side of the surface as a light, useful for closed meshes whose interior is not visible.
Back::	Treat only the back side of the surface as a light for sampling.
Front and Back::	Treat surface as a light for sampling, emitting from both the front and back side.

Transparent Shadows

Use transparent shadows if it contains a Transparent BSDF, disabling will render faster but will not give accurate shadows.

Bump Map Correction

Applies corrections to solve shadow terminator artifacts caused by bump mapping.

Volume

Sampling Method

Distance::	For dense volumes lit from far away *Distance* sampling is usually more efficient.
Equiangular::	If you have got a light inside or near the volume then *equiangular* sampling is better.
Multiple Importance::	If you have a combination of both, then the multiple importance sampling will be better.

Interpolation

Interpolation method to use for the volume objects and smoke simulation grids.

Linear::	Simple interpolation which gives good results for thin volumes.

Cubic::	Smoothed high-quality interpolation needed for more dense volumes, but slower.

Homogeneous

Assume volume has the same density everywhere (not using any textures), for faster rendering. For example absorption in a glass object would typically not have any textures, and so the renderer can be set to avoid taking small steps to sample the volume shader. Usually this is automatically determined by the renderer. This setting provides a manual control for cases where it is not detected.

Step Rate

Adjust distance between volume shader samples for volume shaders. This is typically used to reduce the step size for procedural shaders that add more detail with procedural textures, when it is not captured by the default step size. See Volume Render Settings for more information.

Light Settings

Reference

Panel:: *Properties* ‣ *Light* and *Shader Editor* ‣ *Sidebar* ‣ *Options* ‣ *Light*

Next to lighting from the background and any object with an emission shader, lights are another way to add light into the scene. The difference is that they are not directly visible in the rendered image, and can be more easily managed as objects of their own type.

Common

Light settings for all renderers.

Cycles

Max Bounces
: Maximum number of times light from the light is allowed to Bounce. Limited by scene-wide bounce settings.

Cast Shadow
: By disabling this option, light from lights will not be blocked by objects in between. This can speed up rendering by not having to trace rays to the light source.

Multiple Importance Sample
: By default lights use only direct light sampling. For area lights and sharp glossy reflections, however, this can be noisy, and enabling this option will enable indirect light sampling to be used in addition to reduce noise.

Shadow Caustics

Mark a light as a refractive caustic caster. This setting can be used in conjunction with the Cast and Receive caustics object settings to selectively speed up refractive caustic rendering of select objects.

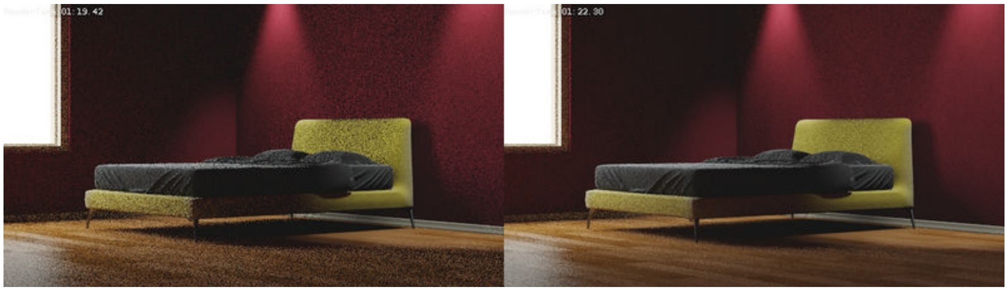
Area Lights

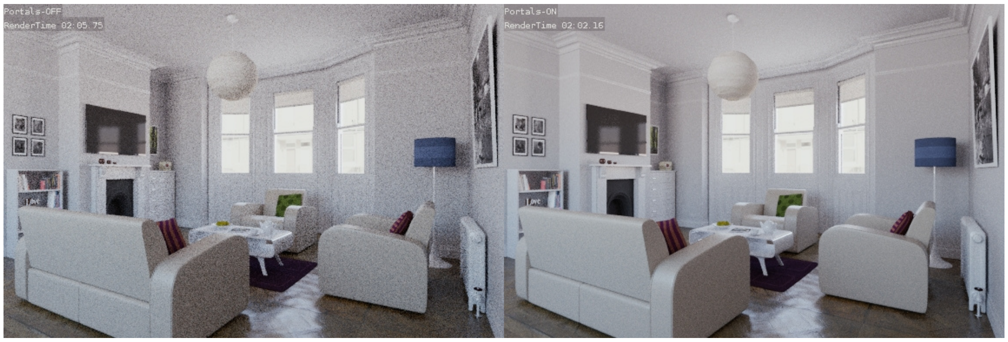
Portals

Area lights can also function as light portals to help sample the environment light, and significantly reduce noise in interior scenes. Note that rendering with portals is usually slower, but as it converges more quickly, less samples are required.

Light portals work by enabling the *Portal* option, and placing areas lights in windows, door openings, and any place where light will enter the interior.

In outdoor scenes most rays do not bounce much and just fly off into the sky and therefore, light portals are not helpful for outdoor scenes.



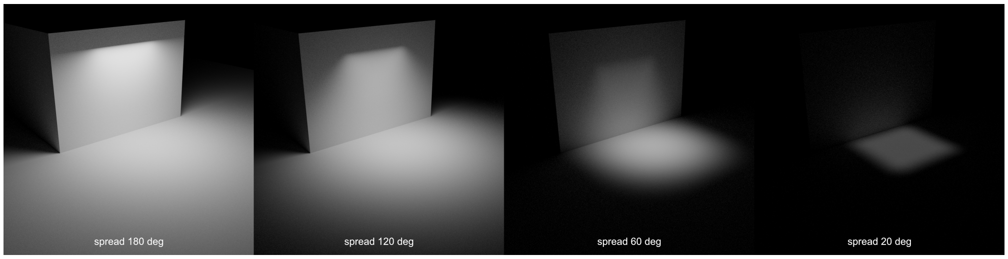


White Room model by Jay Hardy.

Beam Shape

Spread

How wide the emitted light fans out controlling how diffused the light is. Larger values create soft shadows while smaller values create sharper light simulating a gridded softbox.


spread 180 deg
spread 120 deg
spread 60 deg
spread 20 deg

Example of Spread at different angles.

GPU Rendering

GPU rendering makes it possible to use your graphics card for rendering, instead of the CPU. This can speed up rendering because modern GPUs are designed to do quite a lot of number crunching. On the other hand, they also have some limitations in rendering complex scenes, due to more limited memory, and issues with interactivity when using the same graphics card for display and rendering.

To enable GPU rendering, go into the *Preferences* ▸ *System* ▸ *Cycles Render Devices*, and select either *CUDA*, *OptiX*, *HIP*, *oneAPI*, or *Metal*. Next, you must configure each scene to use GPU rendering in *Properties* ▸ *Render* ▸ *Device*.

Rendering Technologies

Blender supports different technologies to render on the GPU depending on the particular GPU manufacturer and operating system.

CUDA – NVIDIA

CUDA is supported on Windows and Linux and requires a Nvidia graphics cards with compute capability 3.0 and higher. To make sure your GPU is supported, see the list of Nvidia graphics cards with the compute capabilities and supported graphics cards.

OptiX – NVIDIA

OptiX is supported on Windows and Linux and requires a Nvidia graphics cards with compute capability 5.0 and higher and a driver version of at least 495.89. To make sure your GPU is supported, see the list of Nvidia graphics cards.

OptiX takes advantage of hardware ray-tracing acceleration in RTX graphics cards, for improved performance.

GPU acceleration for OpenImageDenoise is available for compute capability 7.0 and higher, which includes all NVIDIA RTX cards.

HIP – AMD

HIP is supported on Windows and Linux and requires a AMD graphics card with the RDNA1 architecture or newer. Both discrete GPUs and APUs are supported.

Supported GPUs include:

- Radeon RX 5000 Series
- Radeon RX 6000 Series
- Radeon RX 7000 Series
- Radeon Pro W6000 Series

Minimum driver versions:

- Windows: Radeon Software 21.12.1 or Radeon PRO Software 21.Q4
- Linux: Radeon Software 22.10 or ROCm 5.3

Please refer to AMD's website for more information about AMD graphics cards and their architectures.

Experimental hardware ray-tracing support is available with the most recent drivers. This can be enabled in the preferences. However there are currently known issues regarding motion blur, hair and point cloud rendering, and degenerate triangle shapes.

GPU accelerated denoising is available on discrete Radeon RX 6000 and Radeon RX 7000 GPUs.

oneAPI – Intel

oneAPI is a computation library that is supported on Windows and Linux and requires a Intel® Arc™ graphics card with the Xe HPG architecture. Hardware acceleration for ray-tracing and denoising is supported.

Supported GPUs include:

- Intel® Arc™ A-Series

Minimum driver versions:

- Windows: Intel Graphics Driver XX.X.101.5518
- Linux: `intel-level-zero-gpu` package 1.3.27642, typically available through the `intel-compute-runtime` package XX.XX.27642.38

Please refer to Intel’s website for more information about Intel graphics cards and their architectures.

GPU accelerated denoising is available on all supported GPUs.

Metal – Apple (macOS)

Metal is supported on Apple computers with Apple Silicon. macOS 13.0 or newer is required to support all features.

GPU accelerated denoising is available on Apple Silicon.

Limitations

- Path Guiding is not supported on any GPU.
- Open Shading Language is only supported for OptiX, with some limitations listed in the documentation.

Frequently Asked Questions

Why is Blender unresponsive during rendering?

On older GPU generations, graphics cards can only either render or draw the user interface. This can make Blender unresponsive while it is rendering. Heavy scenes can also make Blender unresponsive on newer GPUs, when using a lot of memory or executing expensive shaders, however this is generally less of a problem.

The only complete solution for this is to use a dedicated GPU for rendering, and another for display.

Why does a scene that renders on the CPU not render on the GPU?

There may be multiple causes, but the most common one is that there is not enough memory on your graphics card. Typically, the GPU can only use the amount of memory that is on the GPU (see Would multiple GPUs increase available memory? for more information). This is usually much smaller than the amount of system memory the CPU can access. With CUDA, OptiX, HIP and Metal devices, if the GPU memory is full Blender will automatically try to use system memory. This has a performance impact, but will usually still result in a faster render than using CPU rendering.

Can multiple GPUs be used for rendering?

Yes, go to *Preferences* ‣ *System* ‣ *Compute Device Panel*, and configure it as you desire.

Would multiple GPUs increase available memory?

Typically, no, each GPU can only access its own memory.

The exception is NVIDIA GPUs connected with NVLink, where multiple GPUs can share memory at a small performance cost. This is can be enabled with Distributed Memory Across Devices in the preferences.

What renders faster?

This varies depending on the hardware used. Different technologies also have different compute times depending on the scene tested. For the most up to date information on the performance of different devices, browse the Blender Open Data resource.

Error Messages

In case of problems, be sure to install the official graphics drivers from the GPU manufacturers website, or through the package manager on Linux. The graphics drivers provided by the computer manufacturer can sometimes be outdated or incomplete.

Error: Out of memory

This usually means there is not enough memory to store the scene for use by the GPU.

Note

One way to reduce memory usage is by using smaller resolution textures. For example, 8k, 4k, 2k, and 1k image textures take up respectively 256MB, 64MB, 16MB and 4MB of memory.

The NVIDIA OpenGL driver lost connection with the display driver

If a GPU is used for both display and rendering, Windows has a limit on the time the GPU can do render computations. If you have a particularly heavy scene, Cycles can take up too much GPU time. Reducing Tile Size in the Performance panel may alleviate the issue, but the only real solution is to use separate graphics cards for display and rendering.

Another solution can be to increase the time-out, although this will make the user interface less responsive when rendering heavy scenes. Learn More Here.

Unsupported GNU version

On Linux, depending on your GCC version you might get this error. See the Nvidia CUDA Installation Guide for Linux for a list of supported GCC versions. There are two possible solutions to this error:

Use an alternate compiler

If you have an older GCC installed that is compatible with the installed CUDA toolkit version, then you can use it instead of the default compiler. This is done by setting the `CYCLES_CUDA_EXTRA_CFLAGS` environment variable when starting Blender.

Launch Blender from the command line as follows:

```
CYCLES_CUDA_EXTRA_CFLAGS="-ccbin gcc-x.x" blender
```

(Substitute the name or path of the compatible GCC compiler).

Remove compatibility checks

If the above is unsuccessful, delete the following line in `/usr/local/cuda/include/host_config.h`:

```
#error -- unsupported GNU version! gcc x.x and up are not
supported!
```

This will allow Cycles to successfully compile the CUDA rendering kernel the first time it attempts to use your GPU for rendering. Once the kernel is built successfully, you can launch Blender as you normally would and the CUDA kernel will still be used for rendering.

CUDA Error: Kernel compilation failed

This error may happen if you have a new NVIDIA graphics card that is not yet supported by the Blender version and CUDA toolkit you have installed. In this case Blender may try to dynamically build a kernel for your graphics card and fail.

In this case you can:

1. Check if the latest Blender version (official or experimental builds) supports your graphics card.
2. If you build Blender yourself, try to download and install a newer CUDA developer toolkit.

Normally users do not need to install the CUDA toolkit as Blender comes with precompiled kernels.

Experimental Features

Reference

Panel:: *Render*

Experimental features are disabled / hidden by default, but can be enabled by setting *Feature Set* to *Experimental* in the Render properties. Enabling the *Experimental Feature Set* will use experimental and incomplete features that might be broken or change in the future.

Adaptive subdivision is currently the only experimental feature.

Render Baking

Cycles shaders and lighting can be baked to image textures. This has a few different purposes, most commonly:

- Baking textures like base color or normal maps for export to game engines.
- Baking ambient occlusion or procedural textures, as a base for texture painting or further edits.
- Creating light maps to provide global illumination or speed up rendering in games.

Setup

Baking requires a mesh to have a UV map, and either a Color Attribute or an Image Texture node with an image to be baked to. The Active Image Texture node or Color Attribute is used as the baking target.

Use Render Bake in intensive light/shadow solutions, such as AO or soft shadows from area lights. If you bake AO for the main objects, you will not have to enable it for the full render, saving render time. Cycles uses the render settings (samples, bounces, …) for baking. This way the quality of the baked textures should match the result you get from the rendered scene.

Settings

Reference

Panel::	*Render ‣ Bake*

Bake
    Perform the baking operation.

Bake from Multires

Bake directly from multires object.

Bake Type

Type of pass to bake.

Combined::	Bakes all materials, textures, and lighting except specularity. The passes that contribute to the combined pass can be toggled individually to form the final map.
Ambient Occlusion::	Bakes ambient occlusion as specified in the World panels. Ignores all lights in the scene.
Shadow::	Bakes shadows and lighting.
Normal::	Bakes normals to an RGB image.
UV::	Mapped UV coordinates, used to represent where on a mesh a texture gets mapped too. This is represented through the red and green channels of the image, the blue channel is encoded with a constant value of 1 but does not hold any information.
Roughness::	Bakes the roughness pass of a material.
Emit::	Bakes Emission, or the Glow color of a material.
Environment::	Bakes the environment (i.e. the world surface shader defined for the scene) onto the selected object(s) as seen by rays cast from the world origin.
Diffuse::	Bakes the diffuse pass of a material.
Glossy::	Bakes the glossiness pass of a material.
Transmission::	Bakes the transmission pass of a material.

View From

Source of reflection ray directions.

Above Surface::	Cast rays from above the surface.
Active Camera::	Use the active camera’s position to cast rays.

Influence

Combined

Lighting

- Direct
  - Add direct lighting contribution.
- Indirect
  - Add indirect lighting contribution.

Contributions

- Diffuse
  - Add diffuse contribution.
- Glossy
  - Add glossy contribution.
- Transmission
  - Add transmission contribution.
- Ambient Occlusion
  - Add ambient occlusion contribution.
- Emit
  - Add emission contribution.

Diffuse, Glossy, Transmission

Contributions

- Direct
  - See above.
- Indirect
  - See above.
- Color
  - Colorize the pass.

- If only *Color* is selected you get the pass color, which is a property of the surface and independent of sampling refinement.
- If *Color* is not selected, you get the direct and/or indirect contributions in gray-scale.
- If *Color* and either *Direct* or *Indirect* are selected, you get the direct and/or indirect contributions colored.

Normal

Space

Normals can be baked in different spaces:

For materials, the same spaces can be chosen in the image texture options next to the existing *Normal Map* setting. For correct results, the setting here should match the setting used for baking.

Object::	Normals in object coordinates, independent of object transformation, but dependent on deformation.
Tangent::	Normals in tangent space coordinates, independent of object transformation and deformation. This is the default, and the right choice in most cases, since then the normal map can be used for animated objects too.

Swizzle R, G, B

Axis to bake into the red, green and blue channel.

Selected to Active

Bake shading on the surface of selected objects to the active object. The rays are cast from the low-poly object inwards towards the high-poly object. If the high-poly object is not entirely involved by the low-poly object, you can tweak the rays start point with *Max Ray Distance* or *Extrusion* (depending on whether or not you are using cage). For even more control you can use a *Cage Object*.

Note

There is a CPU fixed memory footprint for every object used to bake from. In order to avoid crashes due to lack of memory, the high-poly objects can be joined before the baking process.

Cage

Cast rays to active object from a cage. A cage is a ballooned-out version of the low-poly mesh created either automatically (by adjusting the ray distance) or manually (by specifying an object to use). When not using a cage the rays will conform to the mesh normals. This produces glitches on the edges, but it is a preferable method when baking into planes to avoid the need of adding extra loops around the edges.

Cage Object

Object to use as cage instead of calculating the cage from the active object with the *Cage Extrusion*.

Cage Extrusion

Distance to use for the inward ray cast when using *Selected to Active* and *Cage*. The inward rays are cast from a version of the active object with disabled Edge Split Modifiers. Hard splits (e.g. when the Edge Split Modifier is applied) should be avoided because they will lead to non-smooth normals around the edges.

Note

When the base mesh extruded does not give good results, you can create a copy of the base mesh and modify it to use as a *Cage*. Both meshes need to have the same Topology (number of faces and face order).

Max Ray Distance

Distance to use for the inward ray cast when using *Selected to Active*. Ray distance is only available when not using *Cage*.

Output

Target

Where to output the baked map.

Image Textures::	Bake to the image data-block associated with the Active Image Texture node. Clear Image If selected, clears the image before baking render.
Active Color Attribute::	Bake to the Active Color Attributes layer on the active mesh. Note, the active object must be a mesh as other object types do not have Color Attributes.

Margin

When baking to images, by default a margin is generated around UV “islands”. This is important to avoid discontinuities at UV seams, due to texture filtering and mip-mapping.

Type

Method to generate the margin.

Extend::	Extend border pixels outwards.
Adjacent Faces::	Fill margin using pixels from adjacent faces across UV seams.

Size

Size of the margin in pixels.

Optimizing Renders

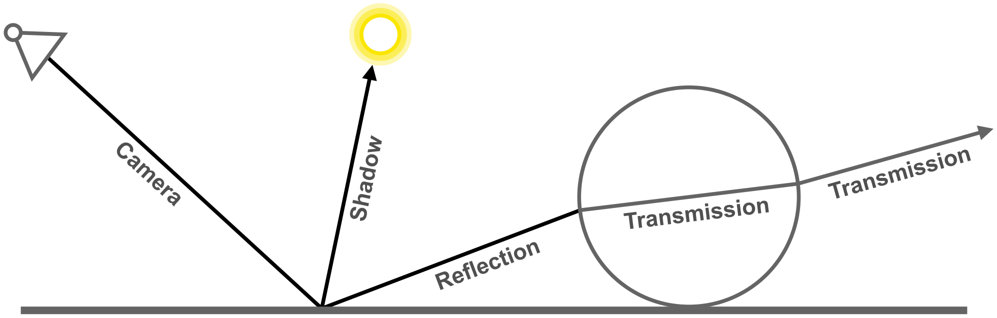
- Reducing Noise
    - Path Tracing
    - The Source of the Noise
    - Bounces
    - Caustics and Filter Glossy
    - Light Falloff
    - Multiple Importance Sampling
    - Glass and Transparent Shadows
    - Light Portals
    - Denoising
    - Clamp Fireflies
- Shader Nodes
    - Node Optimizations
    - Run-Time Optimizations
    - Open Shading Language

Reducing Noise

When performing a final render, it is important to reduce noise as much as possible. Here we will discuss a number of tricks that, while breaking the laws of physics, are particularly important when rendering animations within a reasonable time. Click to enlarge the example images to see the noise differences well.

Path Tracing

Cycles uses path tracing with next event estimation, which is not good at rendering all types of light effects, like caustics, but has the advantage of being able to render more detailed and larger scenes compared to some other rendering algorithms. This is because we do not need to store, for example, a photon map in memory, and because we can keep rays relatively coherent to use an on-demand image cache, compared to e.g. bidirectional path tracing.



We do the inverse of what reality does, tracing light rays from the camera into the scene and onto lights, rather than from the light sources into the scene and then into the camera. This has the advantage that we do not waste light rays that will not end up in the camera, but also means that it is difficult to find some light paths that may contribute a lot. Light rays will be sent either according to the surface BSDF, or in the direction of known light sources.

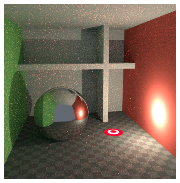
See also

For more details, see the Light Paths and Sampling documentation.

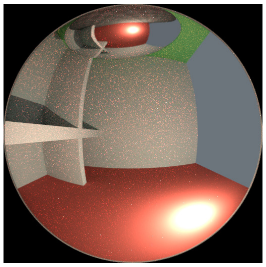
The Source of the Noise

To understand where noise can come from, take for example the scene below. When we trace a light ray into the location marked by the white circle on a red dot, the second image below gives an impression of what the diffuse shader "sees".

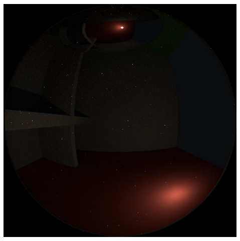
To find the light that is reflected from this surface, we need to find the average color from all these pixels. Note the glossy highlight on the sphere, and the bright spot the light casts on the nearby wall. These hot-spots are much brighter than other parts of the image and will contribute significantly to the lighting of this pixel.



The scene.



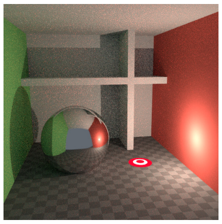
Irradiance at the shading point.



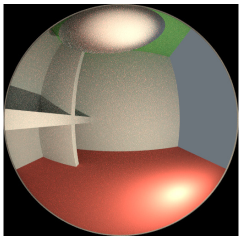
The detected highlights.

The light is a known light source, so its location is already known, but the glossy highlight(s) that it causes are a different matter. The best we can do with path tracing is to distribute light rays randomly over the hemisphere, hoping to find all the important bright spots. If for some pixels we miss some bright spot, but we do find it for another, that results in noise. The more samples we take, the higher the probability that we cover all the important sources of light.

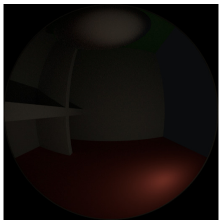
With some tricks we can reduce this noise. If we blur the bright spots, they become bigger and less intense, making them easier to find and less noisy. This will not give the same exact result, but often it's close enough when viewed through a diffuse or soft glossy reflection. Below is an example of using Glossy Filter and Light Falloff.



Using Glossy Filter & Light Falloff.



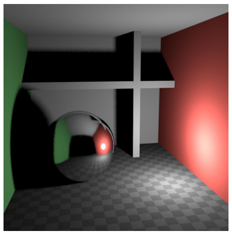
Irradiance at the shading point.

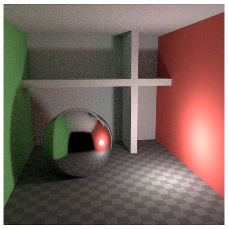


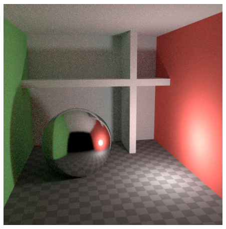
The detected highlights.

Bounces

In reality light will bounce a huge number of times due to the speed of light being very high. In practice more bounces will introduce more noise, and it might be good to use something like the Limited Global Illumination preset in the Light Paths Section that uses *fewer* bounces for different shader types. Diffuse surfaces typically can get away with fewer bounces, while glossy surfaces need a few more, and transmission shaders such as glass usually need the most.


No bounces.

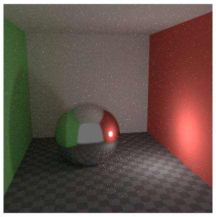

Two bounces at max.

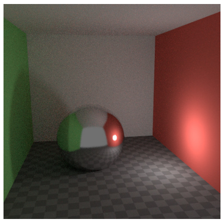

Four bounces at max.

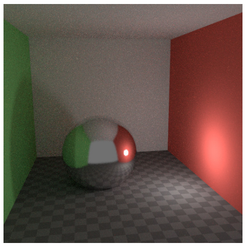
Also important is to use shader colors that do **not** have components of value 1.0 or values near that; try to keep the maximum value to 0.8 or less and make your lights brighter. In reality, surfaces are rarely perfectly reflecting all light, but there are of course exceptions; usually glass will let most light through, which is why we need more bounces there. High values for the color components tend to introduce noise because light intensity then does not decrease much as it bounces off each surface.

Caustics and Filter Glossy

Caustics are a well-known source of noise, causing Fireflies. They happen because the renderer has difficulty finding specular highlights viewed through a soft glossy or diffuse reflection. There is a No Caustics option to disable glossy behind a diffuse reflection entirely. Many renderers will typically disable caustics by default.


Default settings.


Caustics disabled.

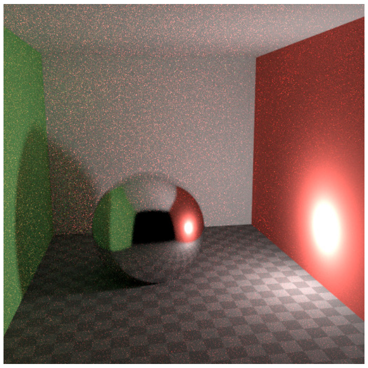

Filter Glossy greater than zero.

However, using *No Caustics* will result in missing light, and it still does not cover the case where a sharp glossy reflection is viewed through a soft glossy reflection. There is a Filter Glossy option to reduce the noise from such cases at the cost of accuracy. This will blur the sharp glossy reflection to make it easier to find, by increasing the shader Roughness.

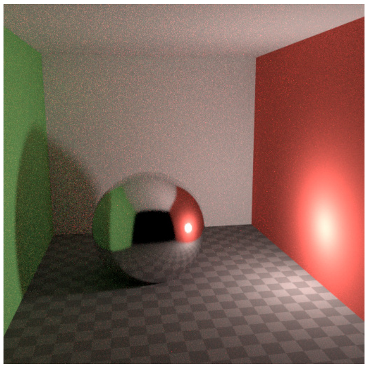
The above images show default settings, no caustics, and filter glossy set to 1.0.

Light Falloff

In reality light in a vacuum will always fall off at a rate of 1/(distance^2). However, as distance goes to zero, this value goes to infinity and we can get very bright spots in the image. These are mostly a problem for indirect lighting, where the probability of hitting such a small but extremely bright spot is low and so happens only rarely. This is a typical recipe for Fireflies.



Hard Falloff.



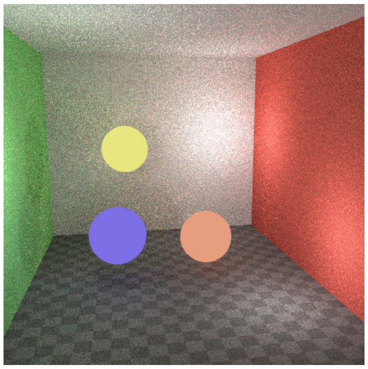
Soft Falloff.

To reduce this problem, the Light Falloff node has a *Smooth factor*, that can be used to reduce the maximum intensity a light can contribute to nearby surfaces. The images above show default falloff and smooth value 1.0.

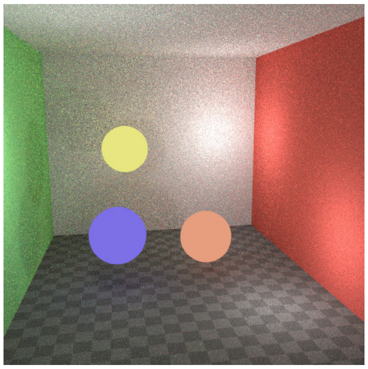
Multiple Importance Sampling

Materials with emission shaders can be configured to use Multiple Importance Sampling (Material Settings). This means that they will get rays sent directly towards them, rather than ending up there based on rays randomly bouncing around. For very bright mesh light sources, this can reduce noise significantly. However, when the emission is not particularly bright, this will take samples away from other brighter light sources for which it is important to find them this way.

The optimal setting here is difficult to guess; it may be a matter of trial and error, but often it is clear that a somewhat glowing object may be only contributing light locally, while a mesh light used as a light would need this option enabled. Here is an example where the emissive spheres contribute little to the lighting, and the image renders with slightly less noise by disabling Multiple Importance on them.



Multiple Importance off.

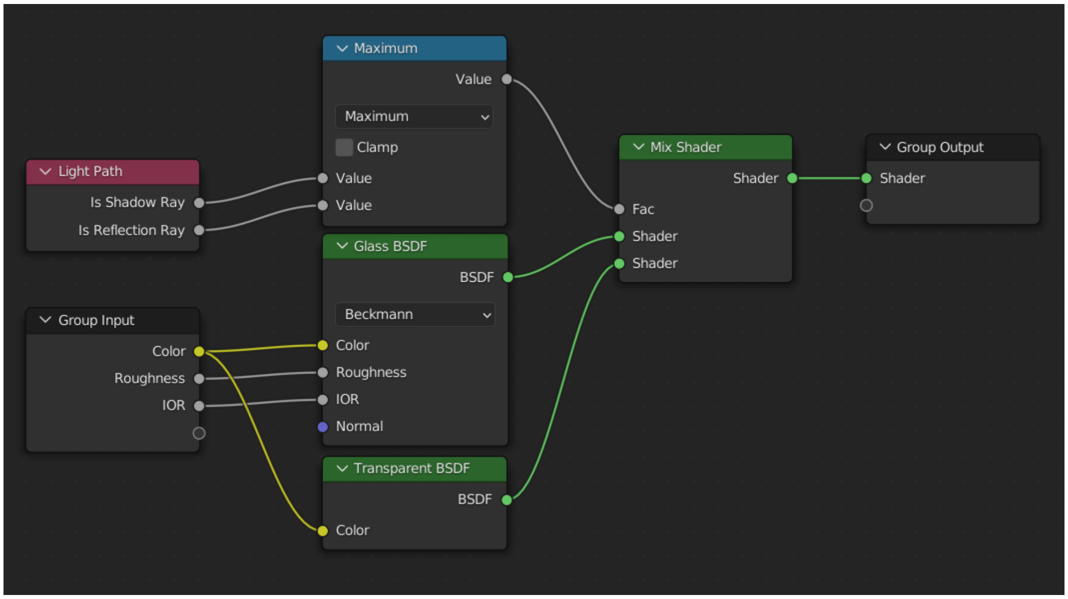


Multiple Importance on.

The world background also has a *Multiple Importance* (Settings) option. This is mostly useful for environment maps that have small bright spots in them, rather than being smooth. This option will then, in a preprocess, determine the bright spots, and send light rays directly towards them. Again, enabling this option may take samples away from more important light sources if it is not needed.

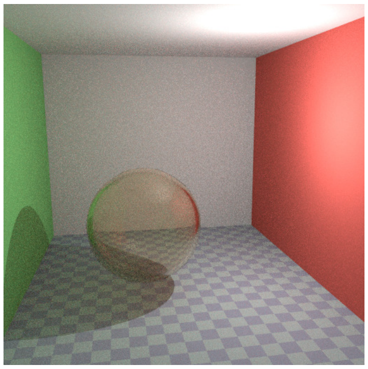
Glass and Transparent Shadows

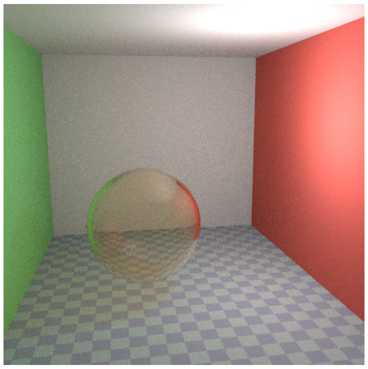
With caustics disabled, glass shadows may appear too dark, and with filter glossy the caustics might be too soft. We can make a glass shader that will use a Glass BSDF when viewed *directly*, and a Transparent BSDF when viewed *indirectly*. The Transparent BSDF can be used for transparent shadows to find light sources straight through surfaces, and will give properly-colored shadows, but without the caustics. The Light Path node is used to determine when to use which of the two shaders.


Maximum
Value
Maximum
Clamp
Value
Value
Light Path
Is Shadow Ray
Is Reflection Ray
Mix Shader
Shader
Fac
Shader
Shader
Group Output
Shader
Glass BSDF
BSDF
Beckmann
Color
Roughness
IOR
Normal
Group Input
Color
Roughness
IOR
Transparent BSDF
BSDF
Color

Optimized glass shader.

Above we can see the node setup used for the glass transparency trick; on the left the render has dark shadows due to missing caustics, and on the right the render with the trick.

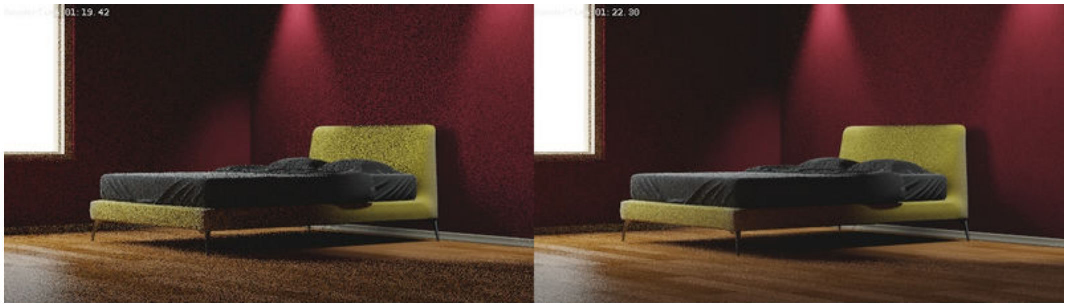


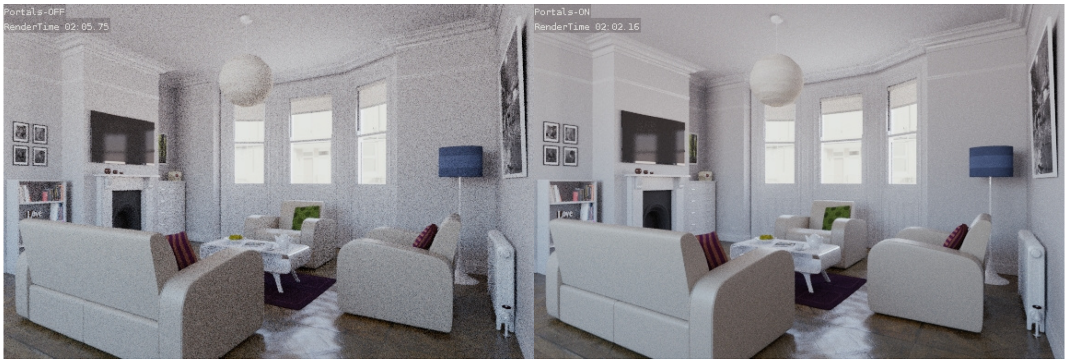


Default Glass BSDF. Optimized Glass Shader.

Light Portals

When rendering a daylight indoor scene where most of the light is coming in through a window or door opening, it is difficult for the integrator to find its way to them. To fix this, use Light Portals. You then will need to modify its shape to match that of the opening that you are trying to fill.

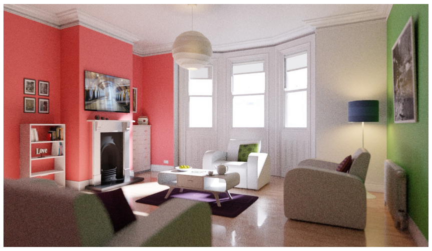

01:19.42
01:22.80


Portals-OFF
RenderTime 02:05.75
Portals-ON
RenderTime 02:02.16

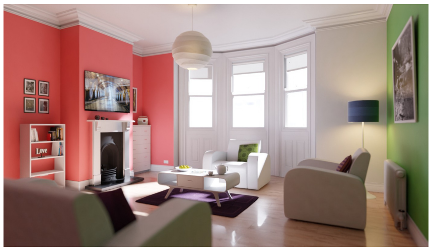
Denoising

Even with all the settings described above there will always end up being some render noise no matter how many samples you use. To fix this there is a post-processing technique to cleanup the final bit of noise. To use this enable Denoising in the *Render* tab of the Properties.

Below is an example render by The Pixelary.



Example render before denoising.

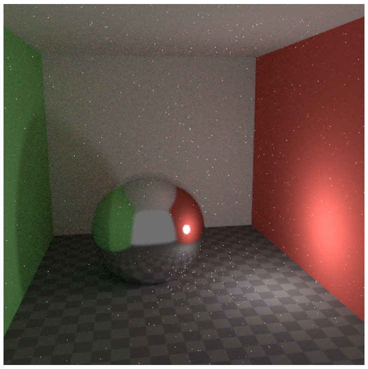


Example render after denoising.

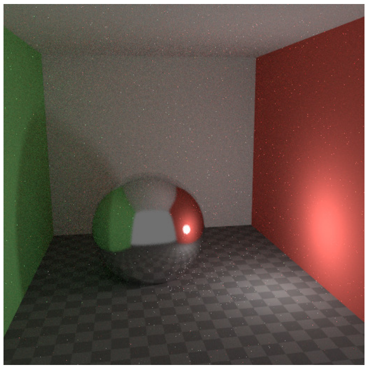
Clamp Fireflies

Ideally with all the previous tricks, Fireflies would be eliminated, but they could still happen. For that, the *intensity* that any individual light ray sample will contribute to a pixel can be *clamped* to a maximum value with the integrator Clamp setting.

If set too low this can cause missing highlights in the image, which might be useful to preserve for camera effects such as bloom or glare. To mitigate this conundrum it's often useful to clamp only indirect bounces, leaving highlights directly visible to the camera untouched.



No Clamp (0).



With Clamp set to 4.

Shader Nodes

Cycles applies a number of shader node optimizations both at compile time and run-time. By exploiting them it is possible to design complicated “Uber Shader” style node groups that incur minimal render time overhead for unused features.

Node Optimizations

As the first step in preparing a node shader for execution, Cycles expands all node groups, as if using the Ungroup tool, and discards UI only features like frames and reroute nodes.

After that, it applies some obvious transformations, for example, it can (the list is not exhaustive):

- Replace the following nodes with the constant result of their evaluation, if all their inputs are determined to be constant:

  RGB, Value, Mix RGB, Math, Vector Math, RGB to BW, Gamma, Bright Contrast, Invert, Separate/Combine RGB/XYZ/HSV, Blackbody, RGB Curves, Vector Curves, Color Ramps.

- Detect Mix RGB, Math and Vector Math nodes that become no-op (without Clamp) or evaluate to 0 as a result of addition, subtraction, multiplication, division or dot/cross product with a known constant 0 or 1 input, and replace with the appropriate input link or constant result.

- Eliminate Mix RGB Mix (without Clamp) and Mix Shader nodes when Factor is known to be 0 or 1 by replacing with the appropriate input value or link.

- Eliminate no-op Mix RGB (except Color Burn, Color Dodge, Lighten, or enabled Clamp), Invert, RGB Curves and Vector Curves nodes with known zero Factor.

- Eliminate Emission and Background shader nodes that do not emit any light, and Add Shader nodes with one or both input arguments missing.
- Eliminate Bump with constant Height input, using its Normal input or Geometry Normal instead. This is useful for implementing node group inputs that default to normal via routing through a no-op Bump before doing math.
- Replace Attribute nodes of the View Layer type with the evaluated attribute value (it is constant within the whole Render Layer).
- Combine multiple copies of the same node with the same inputs into only one instance.

Finally, any nodes that end up not connected either directly or indirectly to the Output node are removed.

Run-Time Optimizations

When executing shaders, a special optimization is applied to Mix Shader nodes. If Factor evaluates to 0 or 1, any nodes that are only reachable via the unused branch of the mix are not evaluated.

This can substantially reduce the performance cost of combining multiple materials in one shader with a Color Attribute, texture, or other input used as a switch.

Open Shading Language

If Open Shading Language is chosen as the rendering backend, node shaders are translated to OSL code and then compiled and executed by the OSL runtime. In the process it applies its own extensive set of optimizations, both at compile time and run-time.

Open Shading Language can optimize out Script nodes if their outputs are unused or constant, even if their OSL shaders have side effects like debug tracing and message passing, which may be confusing. For that reason message passing with `setmessage` and `getmessage` should generally not be used for passing information forward in the graph; explicitly passing information through sockets should be preferred.

Workbench

- Introduction
- Performance
- Sampling
- Lighting
- Color
- Options
- Grease Pencil
- Viewport Display

Introduction

The Workbench Engine is a render engine optimized for fast rendering during modeling and animation preview. It is not intended to be a render engine that will render final images for a project. Its primary task is to display a scene in the 3D Viewport when it is being worked on.

Note

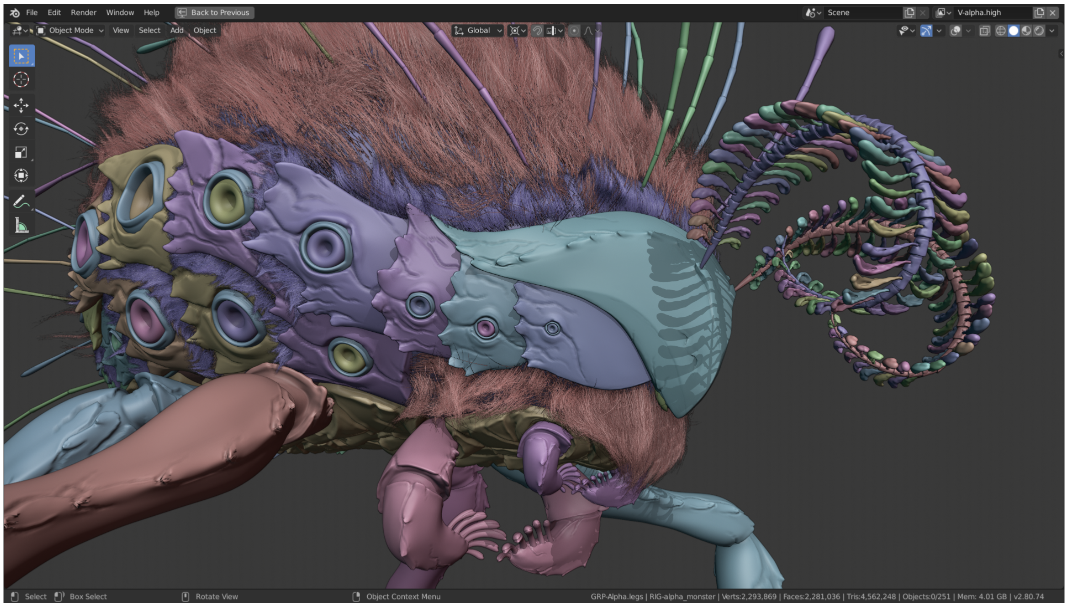
While it is not intended to be used for final renders, the Workbench render engine can be selected as the *Render Engine* in the Render properties.

By default the 3D Viewport uses Workbench to shade and light objects. Unlike other render engines such as EEVEE or Cycles, the Workbench engine does not use shader nodes. Instead, shading settings can be tweaked in the 3D Viewport's Shading popover or the render properties when doing final renders.

Workbench supports assigning random colors to objects to make each visually distinct. Other coloring mechanisms also exist, including; materials, Color Attributes, and textures.

Workbench also has an X-ray mode to see through objects, along with cavity and shadow shading to help display details in objects. Workbench supports several lighting mechanisms including studio lighting and MatCaps.

The image below is an excellent example of the Workbench engine's capabilities using random coloring and shadows to show the details of the model.


File
Edit
Render
Window
Help
Back to Previous
Scene
V-alpha.high
Object Mode
View
Select
Add
Object
Global
Select
Box Select
Rotate View
Object Context Menu

```
GRP-Alpha.legs | RIG-alpha_monster | Verts:2,293,869 | Faces:2,281,036 | Tris:4,562,248 | Objects:0/251 | Mem: 4.01 GB | v2.80.74
```

Workbench example.

Performance

Reference

Panel:: *Properties* ‣ *Render* ‣ *Performance*

High Quality Normals
  Uses higher precision normals and tangents which can improve visual quality for dense meshes with high frequency textures at the cost of memory.

Sampling

The quality of the renders can be adjusted by changing the Anti-Aliasing method. A different one can be selected for the 3D Viewport, viewport rendering and for final rendering.

The setting for the 3D Viewport is a user preference to specify the anti-aliasing method that runs best on the used system. The setting for viewport rendering and for final rendering is saved per scene.

Reference

Panel:: *Render ▸ Sampling Preferences ▸ Viewport*

No Anti-Aliasing
: With this option selected no anti-aliasing will be applied.

Single Pass Anti-Aliasing
: Scene will be rendered with a post-process anti-aliasing pass.

Multisample
: The scene will be rendered multiple times with a slight offset. The anti-aliasing will be gathered from the multiple renders. The number of samples are predefined so it uses the best distribution of the samples.

  5, 8, 11, 16, 32

  Tip

  Multisample anti-aliasing is well suited for rendering small details like hair.

Progressive Viewport Rendering

For the 3D Viewport, one sample is rendered at a time. When there are no changes to the scene or viewport the next sample will be rendered.

Lighting

The Workbench engine does not use the lights of the scene. The lighting conditions that will be used can be set in the Lighting panel.

Reference

Panel::	*Properties* ‣ *Render* ‣ *Lighting*

Flat::	Objects are “shaded” in a flat color, without any hilights or shadows.
Studio::	Use a predefined studio lighting setup, such as a key light shining from the front and a rim light shining from the back. Click the sphere to choose a different setup. The studio lights can be configured in the Preferences. By default, they follow the viewport camera around, but this can be changed: World Space Lighting Keep the lights fixed in place rather than following the viewport camera. Rotation The rotation of the lights on the Z axis.
MatCap::	Use a Material Capture, which is an image with texturing, lighting and even reflections baked into it. Objects are shaded by simply picking colors from this image based on the direction of the normal in relation to the camera. Click the sphere to choose a different MatCap, or the double arrow button to flip it horizontally. Custom MatCaps can be loaded in the Preferences.

Color

The colors that the Workbench uses to render objects can be changed.

Reference

Panel:: *Render* ▸ *Color*

Material:: Use the color that can be set per material in the Viewport Display Material panel.

Object:: Use the color that can be set per object in the Viewport Display Object panel.

Attribute:: Display the active Color Attribute of an object. When an object has no active Color Attribute it will be rendered in the color set in the Viewport Display Object panel.

Single:: Render the whole scene using a single color. The color can be chosen.

Random:: A random color will be selected for every object in the scene.

Texture:: Show the texture from the active Image Texture Node using the active UV map. If there is no active texture, the object will be rendered with the settings in the Material's Viewport Display panel.

Options

Reference

Menu:: *Properties* ‣ *Render* ‣ *Options* panel.

Backface Culling
Use backface culling to hide backsides of faces.

X-Ray
Render the scene transparent. With the slider you can control how transparent the scene should appear.

Shadow
Renders a sharp shadow in the scene.

Darkness
Defines how dark the shadow should be rendered. This slider can be adjusted between 0 (shadow not visible) and 1 (shadow is black).

Light Direction
Controls the direction of the light source that casts the shadows.

Shadow Shift
Controls the Shadow termination angle. It can be used to limit self shadowing artifacts.

Shadow Focus
Controls the falloff near the edge of the shadow.

Cavity
Highlight ridges and valleys in the scene geometry.

Type
How to calculate cavities.

World::	More precise but is slower to calculate.
Screen::	Fast but does not take the size of the ridges and valleys into account.
Both::	Use both methods.

Ridge

Control the visibility of ridges.

Valley

Control the visibility of valleys.

Depth of Field

Use the Depth of Field settings of the active camera in the viewport. Only visible when looking through the camera.

The settings are located on *Properties* ▸ *Camera* ▸ *Depth of Field* panel.

Outline

Render the outline of objects in the viewport. The color of the outline can be adjusted.

Specular Highlighting

Render specular highlights.

Note

Only available when Lighting is set to *Studio* lighting or when a MatCap has been selected that contains a specular pass.

Grease Pencil

Reference

Panel:: *Render ‣ Grease Pencil*

This panel is comprised of settings to control the rendering of Grease Pencil Lines.

Anti-Aliasing Threshold
: Threshold for the edge detection algorithm used to correct aliasing, higher values might over blur some part of the image.

Viewport Display

The Workbench engine does not work with shader trees. In various tabs of the Properties are Viewport Display panels where settings can be adjusted that the Workbench engine uses.

Object

The Viewport Display panel in the Object Properties has several settings that are used by the Workbench Engine.

Reference

Panel::	*Properties* ▸ *Object* ▸ *Viewport Display*

Shadow
: When the *Shadow* in the Options is enabled this object will cast a shadow.

In Front
: When checked the object will be rendered in front of the other objects in the scene.

Color
: The color to render the object in when object color needs to be rendered. The alpha channel can be used to render the object transparent.

Material

The Viewport Display panel in the Material Properties has several settings that are used by the Workbench Engine.

Reference

Panel:: *Properties* ▸ *Material* ▸ *Viewport Display*

Color

The color when rendering the material. The alpha channel can be used to render the object transparent.

Metallic

Changes the amount of specular lighting. This is only available when *Specular Lighting* in the Options is enabled.

Roughness

Changes the amount of roughness for specular lighting. This is only available when *Specular Lighting* in the Options is enabled.

World

The Viewport Display panel in the World Properties has several settings that are used by the Workbench Engine.

Reference

Panel:: *Properties* ▸ *World* ▸ *Viewport Display*

Color

The color of the world background. This color will be rendered in the background of the scene.

Cameras

A camera is an object that provides a means of rendering images from Blender. It defines which portion of a scene is visible in the rendered image.

Cameras are invisible in renders, so they do not have any material or texture settings. However, they do have *Object* and *Editing* setting panels available which are displayed when a camera is the active object.

See also

3D Viewport Camera Navigation for documentation about managing cameras in the viewport.

Properties

Reference

Mode::	Object Mode
Editor::	*Properties* ‣ *Camera*

Lens

Type

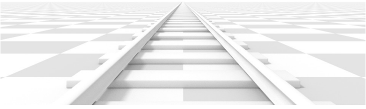
The camera lens options control the way 3D objects are represented in a 2D image.

Perspective

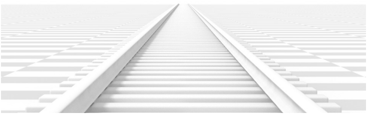
This matches how you view things in the real world. Objects in the distance will appear smaller than objects in the foreground, and parallel lines (such as the rails on a railroad) will appear to converge as they get farther away.

Focal Length/Field of View

The Focal Length controls the amount of zoom, i.e. the amount of the scene which is visible all at once. Longer focal lengths result in a smaller FOV (more zoom), while short focal lengths allow you to see more of the scene at once (larger FOV, less zoom).



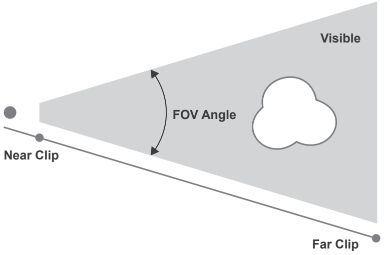
Perspective camera with 35 mm focal length.



Perspective camera with 210 mm focal length instead of 35 mm.

Lens Unit

The focal length can be set either in terms of millimeters or the actual Field of View as an angle.



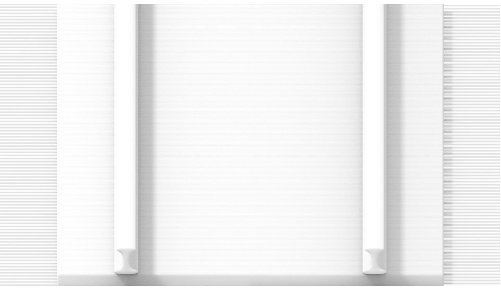
Hint

While the camera is moving towards an object the *Focal Length* property can be decreased to produce a *Dolly Zoom* camera effect, or vice versa.

This video demonstrates the *Dolly Zoom* camera effect.

Orthographic

With *Orthographic* perspective objects always appear at their actual size, regardless of distance. This means that parallel lines appear parallel, and do not converge like they do with *Perspective*.

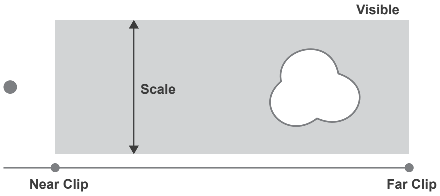


Render from the same camera angle as the previous examples, but with orthographic perspective.

Orthographic Scale

This controls the apparent size of objects projected on the image.

Note that this is effectively the only setting which applies to orthographic perspective. Since parallel lines do not converge in orthographic mode (no vanishing points), the lens shift settings are equivalent to translating the camera in the 3D Viewport.

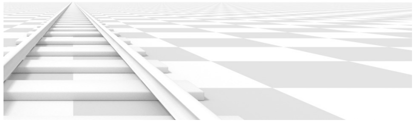


Panoramic

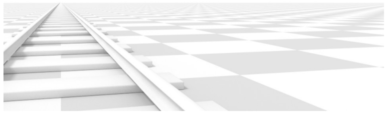
Panoramic cameras only work in Cycles. See the Cycles panoramic camera settings for more information.

Shift

Allows for the adjustment of *vanishing points*. *Vanishing points* refer to the positions to which parallel lines converge. In these render examples, the most obvious vanishing point is at the end of the railroad.



Horizontal lens shift of 0.330.



Rotation of the camera object instead of a lens shift.

Notice how the horizontal lines remain perfectly horizontal when using the lens shift, but do get skewed when rotating the camera object.

Note

Using lens shift is equivalent to rendering an image with a larger FOV and cropping it off-center.

Clip Start and End

The interval in which objects are directly visible. Any objects outside this range still influence the image indirectly, as further light bounces are not clipped.

Note

For viewport rendering, setting clipping distances to limited values is important to ensure sufficient rasterization precision. Ray tracing renders do not suffer from this issue so much, and as such more extreme values can safely be set.

Tip

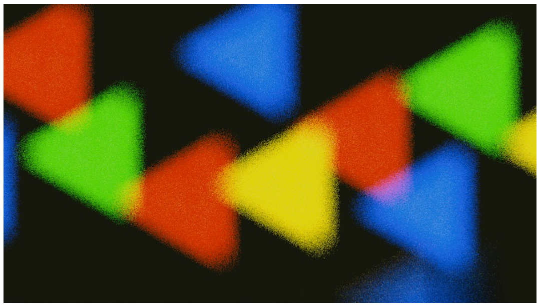
When *Limits* in the *Viewport Display* panel is enabled, the clip bounds will be visible as two yellow connected dots on the camera's line of sight.

See also

- 3D Viewport clipping

Depth of Field

Real-world cameras transmit light through a lens that bends and focuses it onto the sensor. Because of this, objects that are a certain distance away are in focus, but objects in front and behind that are blurred.



Example of DOF bokeh effect.

The area in focus is called the *focal point* and can be set using either an exact value, or by using the distance between the camera and a chosen object:

Focus Object

Choose an object which will determine the focal point. Linking an object will deactivate the distance parameter.

Focal Distance

Sets the distance to the focal point when no *Focus Object* is specified. This distance can be visualized in the 3D Viewport by enabling *Limits* in the camera's Viewport Display panel.

Hint

Use the eyedropper icon or hover the mouse over the *Focal Distance* property and press `E` to enable the depth *Depth Picker*. Then `LMB` on a point in the 3D Viewport to sample the distance from that point to the camera.

Aperture

F-Stop

F-Stop ratio that defines the amount of blurring. Lower values give a strong depth of field effect.

Blades

Total number of polygonal blades used to alter the shape of the blurred objects in the render, and render preview. As with the viewport, the minimum amount of blades to enable the bokeh effect is 3, resulting in a triangular-shaped blur.

Rotation

Rotate the polygonal blades along the facing axis, and will rotate in a clockwise, and counter-clockwise fashion.

Ratio

Change the amount of distortion to simulate the anamorphic bokeh effect. A setting of 1.0 shows no distortion, where a number below 1.0 will cause a horizontal distortion, and a higher number will cause a vertical distortion.

Camera

These settings adjusts properties that relate to a physical camera body. Several Presets can be chosen to match real-world cameras.

Sensor Fit

Adjusts how the camera's sensor fits within the outputs dimension adjusting the angular field of view.

Auto::	Calculates a square sensor size based on the larger of the Resolution dimensions.
Horizontal::	Manually adjust the *Width* of the sensor, the *Height* is calculated based on the aspect ratio of the output's Resolution.
Vertical::	Manually adjust the *Height* of the sensor, the *Width* is calculated based on the aspect ratio of the output's Resolution.

Size / Width, Height

This setting is an alternative way to control the field of view, as opposed to modifying the focal length. It is useful to match a camera in Blender to a physical camera and lens combination, e.g. for motion tracking.

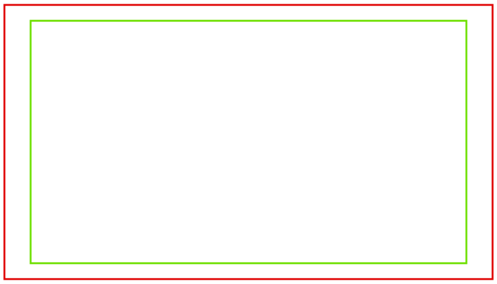
Safe Areas

Safe areas are guides used to position elements to ensure that the most important parts of the content can be seen across all screens.

Different screens have varying amounts of Overscan (especially older TV sets). That means that not all content will be visible to all viewers, since parts of the image surrounding the edges are not shown. To work around this problem TV producers defined two areas where content is guaranteed to be shown: action safe and title safe.

Modern LCD/plasma screens with purely digital signals have no Overscan, yet safe areas are still considered best practice and may be legally required for broadcast.

In Blender, safe areas can be set from the Camera and Sequencer views.



Red line: Action safe. Green line: Title safe.

The Safe Areas can be customized by their outer margin, which is a percentage scale of the area between the center and the render size. Values are shared between the Video Sequence editor and camera view.

Title Safe Margins X/Y
: Also known as *Graphics Safe*. Place all important information (graphics or text) inside this area to ensure it can be seen by the majority of viewers.

Action Safe Margins X/Y
: Make sure any significant action or characters in the shot are inside this area. This zone also doubles as a sort of “margin” for the screen which can be used to keep elements from piling up against the edges.

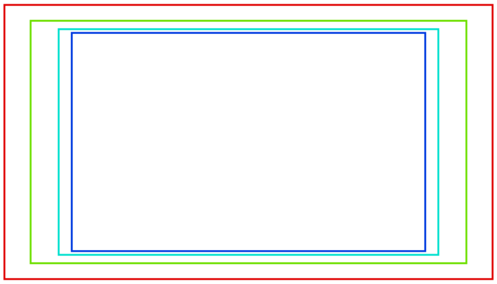
Tip

Each country sets a legal standard for broadcasting. These include, among other things, specific values for safe areas. Blender defaults for safe areas follow the EBU (European Union) standard. Make sure you are using the correct values when working for broadcast to avoid any trouble.

Center-Cut Safe Areas

Center-cuts are a second set of safe areas to ensure content is seen correctly on screens with a different aspect ratio. Old TV sets receiving `16:9` or `21:9` video will cut off the sides. Position content inside the center-cut areas to make sure the most important elements of your composition can still be visible in these screens.

Blender defaults show a `4:3` (square) ratio inside `16:9` (widescreen).



Cyan line: action center safe. Blue line: title center safe.

Background Images

A background picture in your camera can be very helpful in many situations: modeling is obviously one, but it is also useful when painting (e.g. you can have reference pictures of faces when painting textures directly on your model…), or animation (when using a video as background), etc.

Background Source
: The source of the background image.

Image::	Use an external image, image sequence, video file or generated texture.
Movie Clip::	Use one of the Movie Clip data-blocks. Active Clip Display a Movie Clip from the scene’s Active Clip. Render Undistorted Display the background image using undistorted proxies when available. Proxy Render Size

Select between full (non-proxy) display or a proxy size to draw the background image.

See also
To build a proxy, the Movie Clip Editor Proxy settings have to be used. Otherwise the proxy settings here have no effect.

Color Space
The color space the image or video file uses within Blender.

View as Render
Apply the color management settings when displaying this image on the screen.

Opacity
Controls the transparency of the background image.

Depth
Choose whether the image is shown behind all objects, or in front of everything.

Frame Method
Controls how the image is placed in the camera view.

Stretch::	Forces the image dimensions to match the camera bounds (may alter the aspect ratio).
Fit::	Scales the image down to fit inside the camera view without altering the aspect ratio.
Crop::	Scales the image up so that it fills the entire camera view, but without altering the aspect ratio (some of the image will be cropped).

Offset X, Y
Positions the background image using these offsets.

In orthographic views, this is measured in the normal scene units. In the camera view, this is measured relative to the camera bounds (0.1 will offset it by 10% of the view width/height).

Rotation

Rotates the image around its center.

Scale

Scales the image up or down from its center.

Flip

X

Swaps the image around, such that the left side is now on the right, and the right now on the left.

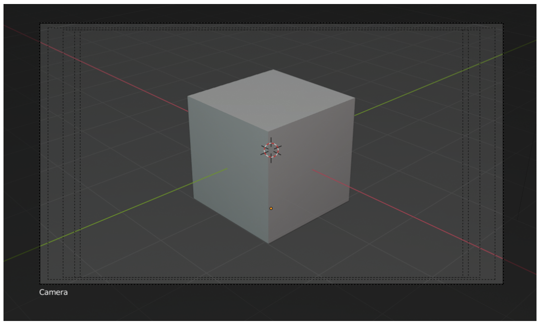
Y

Swaps the image around, such that the top side is now on the bottom, and the bottom now on the top.

Note

Movie Clips or images with view as render are only visible behind objects when film transparency is enabled or the scene world is disabled in the viewport.

Viewport Display



Camera view displaying safe areas, sensor and name.

Size

Size of the camera visualization in the 3D Viewport. This setting has **no** effect on the render output of a camera. The camera visualization can also be scaled using the standard Scale `S` transform key.

Show

Limits

Shows an orange line indicating the *Clip Start* and *End* values, as well as a yellow cross indicating the *Focus Distance*. If the *Focus Distance* gizmo is enabled in the 3D Viewport's gizmo settings, this cross can also be dragged with the mouse to adjust the distance.

Mist

Toggles viewing of the mist limits on and off. The limits are shown as two connected white dots on the camera line of sight. The mist limits and other options are set in the *World* panel, in the Mist section.

Sensor

Displays a dotted frame in camera view.

Name

Toggle name display on and off in camera view.

Composition Guides

Composition Guides enable overlays onto the camera display that can help when framing a shot.

Thirds

Adds lines dividing the frame in thirds vertically and horizontally.

Center

Center

Adds lines dividing the frame in half vertically and horizontally.

Diagonal

Adds lines connecting opposite corners.

Golden
  Ratio
    Divides the width and height into Golden proportions (about 0.618 of the size from all sides of the frame).

  Triangle A
    Displays a diagonal line from the lower left to upper right corners, then adds perpendicular lines that pass through the top left and bottom right corners.

  Triangle B
    Same as A, but with the opposite corners.

Harmony
  Triangle A
    Displays a diagonal line from the lower left to upper right corners, then lines from the top left and bottom right corners to 0.618 the lengths of the opposite side.

  Triangle B
    Same as A, but with the opposite corners.

Passepartout
  This option darkens the area outside of the camera's field of view. The opacity of the passepartout can be adjusted using the value slider.

Tip

If the Passepartout is fully opaque, Blender will make optimizations to speed up the rendering of areas inside the camera view.

Lights

- Light Objects
  - Common Settings
  - Renderer Settings
  - Point Light
  - Spot Light
  - Area Light
  - Sun Light
  - Power of Lights
- Light Linking
  - Setup
  - Include & Exclude
  - Performance
- World Environment
  - Surface
  - Volume
  - Viewport Display

Light Objects

Reference

Panel::	*Properties* ▸ *Light* and *Shader Editor* ▸ *Sidebar* ▸ *Settings*

Common Settings

Type
Defines the type of light.

Color
Color tint of the emitted light.

Renderer Settings

- EEVEE specific settings
- Cycles specific settings

Point Light

The point light is an omni-directional point of light, that is, a point radiating the same amount of light in all directions. It's visualized by a plain, circled dot. Being a point light source, the direction of the light hitting an object's surface is determined by the line joining the light and the point on the surface of the object itself. It can be used as simple model of e.g. a light bulb.

Light intensity/energy decays based on (among other variables) distance from the point light to the object. In other words, surfaces that are further away will be rendered darker.

Power

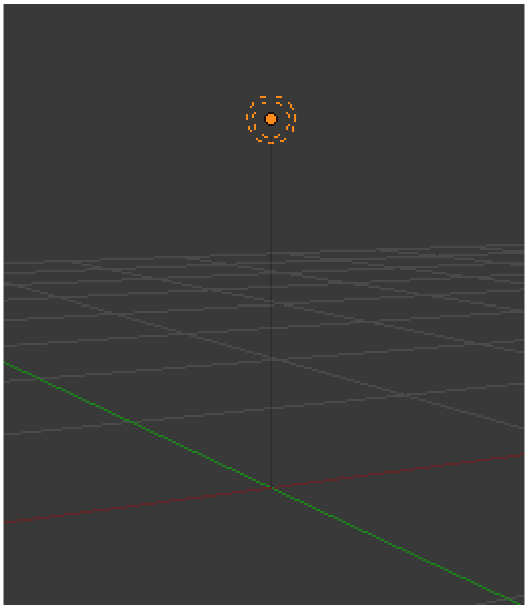
Power of the light in Watts. Higher values increase the intensity of the light. Negative values can be set, but should be avoided for predictable and physically based result.

Soft Falloff

Apply falloff to avoid sharp edges when the light geometry intersects with other objects.

Radius

When larger than zero, light will be emitted from a spherical surfaces with the specified radius. Lights with larger size have softer shadows and specular highlights, and they will also appear dimmer because their power is distributed over a larger area.



Point light.

Spot Light

A spot light emits a cone-shaped beam of light from the tip of the cone, in a given direction.

Power

Power of the light in Watts. Higher values increase the intensity of the light. Negative values can be set, but should be avoided for predictable and physically based result.

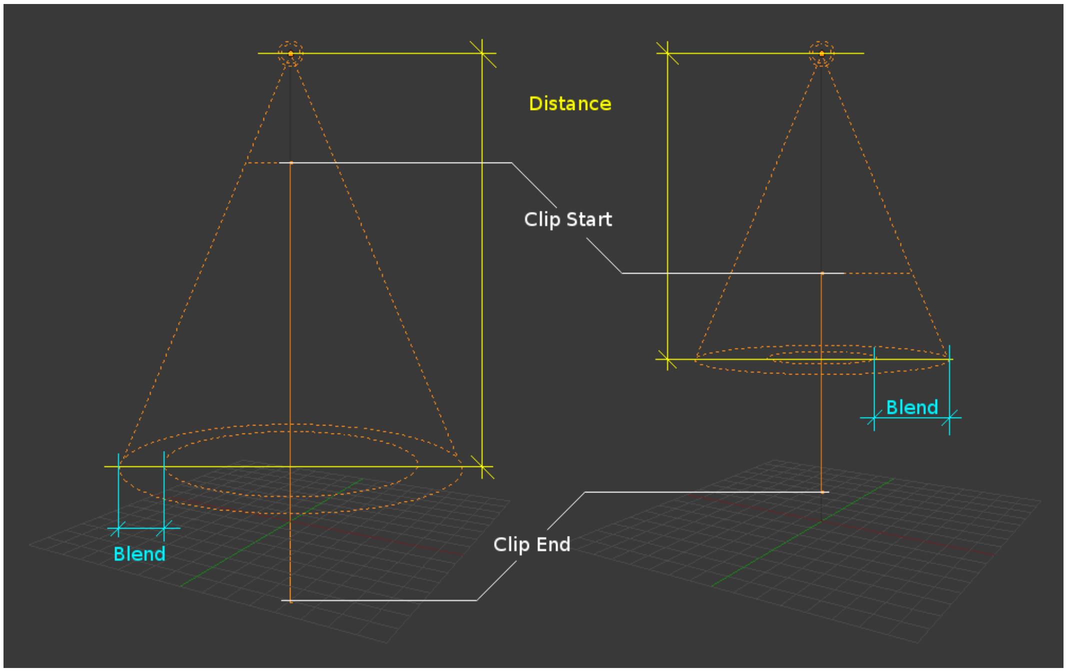
Soft Falloff

Apply falloff to avoid sharp edges when the light geometry intersects with other objects.

Radius

When larger than zero, light will be emitted from a spherical surfaces with the specified radius. Lights with larger size have softer shadows and specular highlights.

Beam/Spot Shape

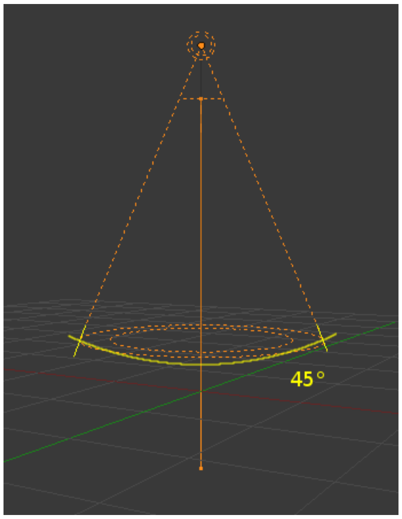


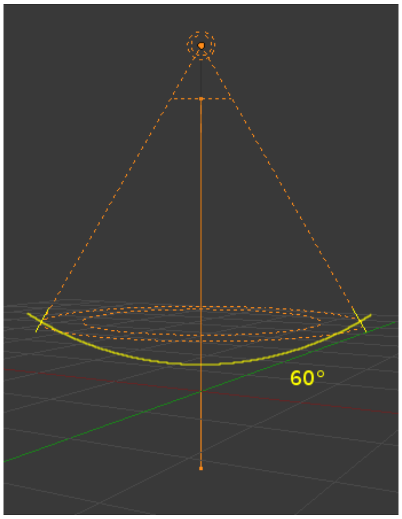
Changing the spot options also changes the appearance of the spotlight as displayed in the 3D Viewport.

Size

The size of the outer cone of a spot, which largely controls the circular area a spot light covers. This slider in fact controls the angle at the top of the lighting cone, and can be between (1.0 to 180.0).

Changing the spot *Size* option.





Blend

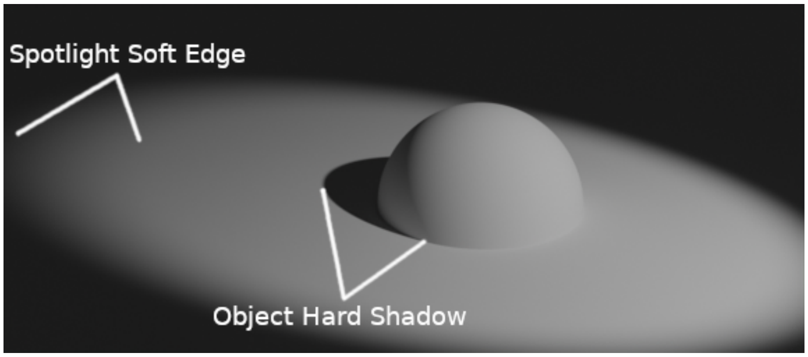
The *Blend* slider controls the inner cone of the spot. The *Blend* value can be between (0.0 to 1.0). The value is proportional and represents that amount of space that the inner cone should occupy inside the outer cone *Size*.

The inner cone boundary line indicates the point at which light from the spot will start to blur/soften; before this point its light will mostly be full strength. The larger the value of *Blend* the more blurred/soft the edges of the spotlight will be, and the smaller the inner cone's circular area will be (as it starts to blur/soften earlier).

To make the spot have a sharper falloff rate and therefore less blurred/soft edges, decrease the value of *Blend*. Setting *Blend* to 0.0 results in very sharp spotlight edges, without any transition between light and shadow.

The falloff rate of the spot light is a ratio between the *Blend* and *Size* values; the larger the circular gap between the two, the more gradual the light fades between *Blend* and *Size*.

Blend and *Size* only control the spot light cone's aperture and softness ("radial" falloff); they do not control the shadow's softness as shown below.



Render showing the soft edge spotlighted area and the sharp/hard object shadow.

Notice in the picture above that the object’s shadow is sharp as a result of the ray tracing, whereas the spotlight edges are soft. If you want other items to cast soft shadows within the spot area, you will need to alter other shadow settings.

Show Cone

Displays a transparent cone in 3D Viewport to visualize which objects are contained in it.

Area Light

The area light simulates light originating from a surface (or surface-like) emitter. For example, a TV screen, office neon lights, a window, or a cloudy sky are just a few types of area light. The area light produces shadows with soft borders by sampling a light along a grid the size of which is defined by the user. This is in direct contrast to point-like artificial lights which produce sharp borders.

Power

Power of the light in Watts. Higher values increase the intensity of the light. Negative values can be set, but should be avoided for predictable and physically based result.

Shape

Shape of the light.

Rectangle::	The shape of the light can be represented as a rectangle and changed with the “X” and “Y” values.
Square::	The shape of the light can be represented as a square and changed with the *Size* property.
Disk::	The shape of the light can be represented as a disk and changed with the *Size* property.
Ellipse::	The shape of the light can be represented as an ellipse and changed with the X and Y values.

Tip

Choosing the appropriate shape for your area light will enhance the believability of your scene. For example, you may have an indoor scene and would like to simulate light entering through a window. You could place a *Rectangular* area light in a window (vertical) or from neon lights (horizontal) with proper ratio for *Size X* and *Size Y*. For the simulation of the light emitted by a TV screen, a vertical *Square* area light would be better in most cases.

Size / Size X / Size Y

Dimensions for the *Square* or *Rectangle*.

Sun Light

A sun light provides light of constant intensity emitted in a single direction from infinitely far away. It can be very handy for a uniform clear daylight open-space illumination. In the 3D Viewport, the sun light is represented by an encircled black dot with rays emitting from it, plus a dashed line indicating the direction of the light.

Note

This direction can be changed by rotating the sun light, like any other object, but because the light is emitted from a location considered infinitely far away, the location of a sun light does not affect the rendered result.

Strength

Strength of the lights in Watts per square meter. Typical values are around 250 for an overcast day and 1000 or more for direct sunlight. See more details at Power of Lights.

Angle

The size of the sun light according to its angular diameter as seen from earth.

Power of Lights

The power of sun lights is specified in Watts per square meter. The power of point lights, spot lights, and area lights is specified in Watts. But this is not the electrical Watts that consumer light bulbs are rated at. It is Radiant Flux or Radiant Power which is also measured in Watts. It is the energy radiated from the light in the form of visible light.

If you want to set the power to real world values, you have to convert the wattage of consumer bulbs or LED lights to radiant flux, but it is not a straightforward process. The wattage of bulbs means the electrical power required to power them. LED lights have a “Watt equivalent” which is neither the electrical power they require nor the amount of light they put out. Some consumer lights specify lumens or luminous flux which is the radiant flux weighted with the wavelengths perceived by the human eye.

To save you from doing the conversion, here is a table of typical power values for point, spot, and area lights:

Real world light	Power	Suggested Light Type
Candle	0.05 W	Point
800 lm LED bulb	2.1 W	Point
1000 lm light bulb	2.9 W	Point
1500 lm PAR38 floodlight	4 W	Area, Disk
2500 lm fluorescent tube	4.5 W	Area, Rectangle
5000 lm car headlight	22 W	Spot, size 125 degrees

And a table of typical Strength values for sun lights:

Sun type	Strength
Clear sky	1000 W/m^2
Cloudy sky	500 W/m^2
Overcast sky	200 W/m^2
Moonlight	0.001 W/m^2

Light Linking

With light linking, lights can be set to affect only specific objects in the scene. Shadow linking additionally gives control over which objects acts as shadow blockers for a light.

This adds more artistic control for lighting by breaking the laws of physics. For example the environment and characters in a shot might have different light setups. A character could have a dedicated linked rim light to make it stand out, and shadow linking could be used to ensure no objects from the environment block it.

Setup

- Select the light or emissive mesh object and go to the Cycles Shading panel or EEVEE Shading panel.
- Create a new light or shadowing linking collection.
- Drag & drop objects or collection from the outliner.

Note

Light linking emissive mesh object is only available for Cycles.

Links can also be set up in the 3D viewport, with the Link Data operator. The active light object is linked to selected receiver or blocker object.

A light or shadow linking collection can be assigned to and shared between multiple light objects. While a scene collection can be directly assigned as a light or shadow linking collection, it is recommended to instead create a dedicated collection and link any scene collection inside it instead. This way it’s easy to include or exclude additional objects without affecting the scene layout.

Include & Exclude

Light receiver objects can be set to be either included or excluded. The behavior is as follows:

- If only included objects are specified, the light only affects those objects.
- If only excluded objects are specified, the light affects all objects in the scene except those specified.
- If both included and excluded objects are specified, the light affects only included objects minus the excluded objects. This can be used to for example set a character collection to be included, and then exclude specific objects part of the character.

Performance

Sampling for light linking is most efficient with the light tree enabled, where a specialized acceleration structure is built for light linking.

When using shadow linking, renders can be slower and trace additional rays, as direct and indirect lighting take different paths.

World Environment

The world defines the environment that the scene is in. The surface shader sets the background and environment lighting, either as a fixed color, sky model or HDRI texture. With volume shaders the entire scene can be covered in mist or other volumetric effects.

Lighting with an HDR image.

Surface

Reference

Panel:: *World ‣ Surface*

The surface shader defines the light emission from the environment into the scene. The world surface is rendered as if it is very distant from the scene, and as such there is no two-way interacting between objects in the scene and the environment, only light coming in. The only shader accepted is the Background node with a color input and strength factor for the intensity of the light.

Image Based Lighting

For image based lighting, use the Environment Texture node rather than the Image Texture node for correct mapping. This supports *Equirectangular* (also known as latitude/longitude) for environment maps, and *Mirror Ball* mapping for converting photos of mirror balls to environment maps.

Volume

Reference

Panel:: *World ▸ Volume*

A volume shader can be applied to the entirely world, filling the entire space.

Currently this is most useful for night time or other dark scenes, as the world surface shader or sun lights will have no effect if a volume shader is used. This is because the world background is assumed to be infinitely far away, which is accurate enough for the sun for example. However, for modeling effects such as fog or atmospheric scattering, it is not a good assumption that the volume fills the entire space, as most of the distance between the sun and the earth is empty space. For such effects it is be better to create a volume object surrounding the scene. The size of this object will determine how much light is scattered or absorbed.

Viewport Display

Reference

Panel:: *World ▸ Viewport Display*

Color

The color to render the 3D Viewport background when choosing World Background.

Materials

- Introduction
  - Setting up Materials
  - Components
  - Physically Based Shading
- Components
  - Surfaces
  - Volumes
  - Displacement
- Assignment
  - Material Slots
  - Reusing Existing Materials
  - Deleting a Material
  - Multiple Materials
- Preview
- Settings
  - Renderer Settings
  - Pass Index
  - Viewport Display
- Line Art
- Legacy Textures
  - Introduction
  - Colors
  - Types

Introduction

Materials control the appearance of meshes, curves, volumes and other objects. They define the substance that the object is made of, its color and texture, and how light interacts with it.

Physically based materials can be created using the Principled BSDF, Principled Hair, and Principled Volume shaders. With these uber shaders, a wide variety of materials including plastic, glass, metal, cloth, skin, hair, smoke and fire can be created.

A flexible shading nodes system is used to set up textures and create entirely different types of materials like toon shading.

Setting up Materials

Materials can be created in either the Material properties, or in the Shader Editor. These provide a different view of the same shader nodes and material settings.

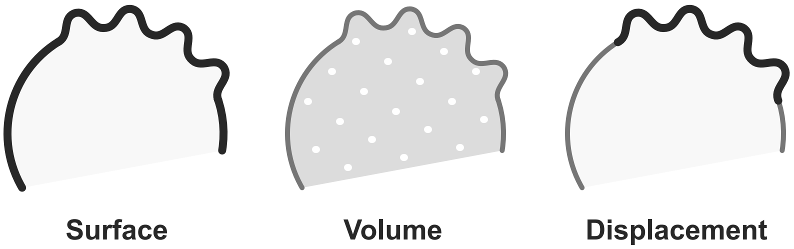
The default Shading workspace has a Shader Editor and a 3D Viewport that can be set to Material Preview or Rendered shading, to interactively preview how the material interacts with objects and lights in the scene.

Materials are data-blocks that can be assigned to one or more objects, and different materials can be assigned to different parts of meshes.

Image textures can be created from scratch in Texture Paint Mode, or by loading in existing images with the Image Texture node. A variety of procedural texture nodes is also available.

Components

Materials consist of three shaders, defining the appearance of the surface, the volume inside the object, and the displacement of the surface.



Surface Shader

The surface shader controls the textures and light interaction at the surface of the mesh.

Volume Shader

The volume shader defines the interior of the mesh. A material can have just a volume shader for cases like smoke and fire, or it can be combined with a surface shader for materials like cloudy glass.

Displacement

The shape of the surface and the volume inside it may be altered by displacement. This way, textures can then be used to make the mesh surface more detailed.

Depending on the settings, the displacement may be virtual, only modifying the surface normals to give the impression of displacement, which is known as bump mapping, or a combination of real and virtual displacement.

Physically Based Shading

The material system is built with physically-based rendering in mind, separating how a material looks and which rendering algorithm is used to render it. This makes it easier to achieve realistic results and balanced lighting, though there are a few things to keep in mind.

In order for materials to work well with global illumination, they should be energy conserving. That means they cannot reflect more light than comes in. This property is not strictly enforced, but if colors are in the range 0.0 to 1.0, and BSDFs are only mixed together with the Mix Shader node, this will automatically be true.

It is however, possible to break this, with color values higher than 1.0 or using the Add Shader node, but one must be careful when doing this to keep materials behaving predictably under various lighting conditions.

Components

- Surfaces
    - Terminology
    - BSDF Parameters
- Volumes
    - Shading
    - Mesh Volumes
    - World Volume
    - Multiple Scattering
- Displacement
    - Bump Only
    - Displacement Only
    - Displacement and Bump

Surfaces

The surface shader defines the light interaction at the surface of the mesh. One or more BSDFs specify if incoming light is reflected back, refracted into the mesh, or absorbed.

Emission defines how light is emitted from the surface, allowing any surface to become a light source.

Terminology

BSDF
  Stands for Bidirectional Scattering Distribution Function. It defines how light is reflected and refracted at a surface.

Reflection
  BSDFs reflect an incoming ray on the same side of the surface.

Transmission
  BSDFs transmit an incoming ray through the surface, leaving on the other side.

Refraction
  BSDFs are a type of *Transmission*, transmitting an incoming ray and changing its direction as it exits on the other side of the surface.

BSDF Parameters

A major difference from non-physically-based renderers is that direct light reflection from lights and indirect light reflection of other surfaces are not decoupled, but rather handled using a single BSDF. This limits the possibilities a bit, but we believe overall it is helpful in creating consistent-looking renders with fewer parameters to tune.

Roughness

For the glossy BSDFs, the *roughness* parameter controls the sharpness of the reflection, from 0.0 (perfectly sharp) to 1.0 (very soft).

Volumes

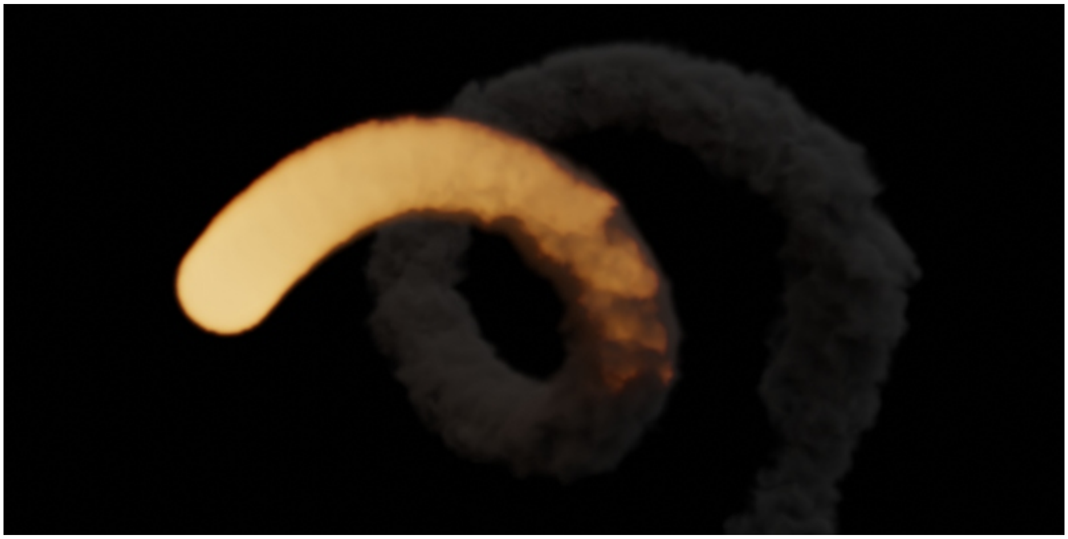
Volume rendering is used to render various effects that cannot be represented by hard surfaces alone.

- Smoke, fire or clouds are set up using a volume object or fluid simulation, with only a volume shader.
- Meshes can also be used to create such shapes by removing the default surface shader and using a volume shader with the mesh shape defining the volume bounds and textures defining the volume density.
- Mist is created with a volume shader for the world, or with a large mesh object encompassing the scene.
- Absorption in glass is simulated by combining a glass surface shader with refraction and a volume absorption shader for the interior of the object.

Shading

Principled Volume

Principled Volume is a physically-based volume shader that can be used to create a wide range of volume materials. It supports scattering, absorption and emission in one easy to use node. Fire can be rendered with blackbody emission.

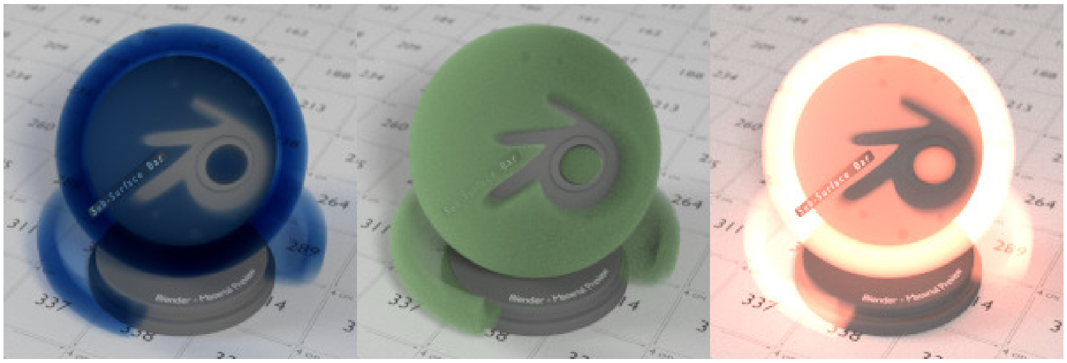


Smoke and fire rendered with Principled Volume shader.

Volume Components

For more control, volume shading components can be manually combined into a custom shader setup.

- Volume Absorption will absorb part of the light as it passes through the volume. This can be used to shade for example black smoke or colored glass objects, or mixed with the Volume Scatter node. This node is similar to the transparent BSDF node, it blocks part of the light and lets other light pass straight through.
- Volume Scatter lets light scatter in other directions as it hits particles in the volume. The anisotropy defines in which direction the light is more likely to scatter. A value of 0 will let light scatter evenly in all directions (similar to the diffuse BSDF node), negative values let light scatter mostly backwards, and positive values let light scatter mostly forward. This can be used to shade white smoke or clouds for example.
- Emission will emit light from the volume, for example for fire.



Volume Absorption, Scatter and Emission

Attributes

When rendering smoke and fire, volume attributes are used to define the shape and shading of the volume. The Principled Volume shader will use them by default, while custom volume shaders can use the Attribute node to get attributes such as density, color and temperature.

Density

All volume shaders have a density input. The density defines how much of the light will interact with the volume, getting absorbed or scattered, and how much will pass straight through. For effects such as smoke you would specify a density field to indicate where in the volume there is smoke and how much (density bigger than 0), and where there is no smoke (density equals 0).

Volumes in the real world consist of particles, a higher density means there are more particles per unit volume. More particles means there is a higher chance for light to collide with a particle and get absorbed or scattered, rather than passing straight through.

Mesh Volumes

Meshes used for volume render should be closed and Manifold. That means that there should be no holes in the mesh. Each edge must be connected to exactly two faces such that there are no holes or T-shaped faces where three or more faces are connected to an edge.

Normals must point outside for correct results. The normals are used to determine if a ray enters or exits a volume, and if they point in a wrong direction, or there is a hole in the mesh, then the renderer is unable to decide what is the inside or outside of the volume.

These rules are the same as for rendering glass refraction correctly.

World Volume

A volume shader can also be applied to the world, filling the entire space.

Currently, this is most useful for night time or other dark scenes, as the world surface shader or sun lights will have no effect if a volume shader is used. This is because the world background is assumed to be infinitely far away, which is accurate enough for the sun for example. However, for modeling effects such as fog or atmospheric scattering, it is not a good assumption that the volume fills the entire space, as most of the distance between the sun and the earth is empty space. For such effects it is be better to create a volume object surrounding the scene. The size of this object will determine how much light is scattered or absorbed.

Multiple Scattering

Real-world effects such as scattering in clouds or subsurface scattering require many scattering bounces. However, unbiased rendering of such effects can be noisy, so by default the number of bounces is zero in Cycles, and no support is available in EEVEE. The effect you get when rendering with zero volume bounces is what is known as “single scattering”, the effect from more bounces is “multiple scattering”.

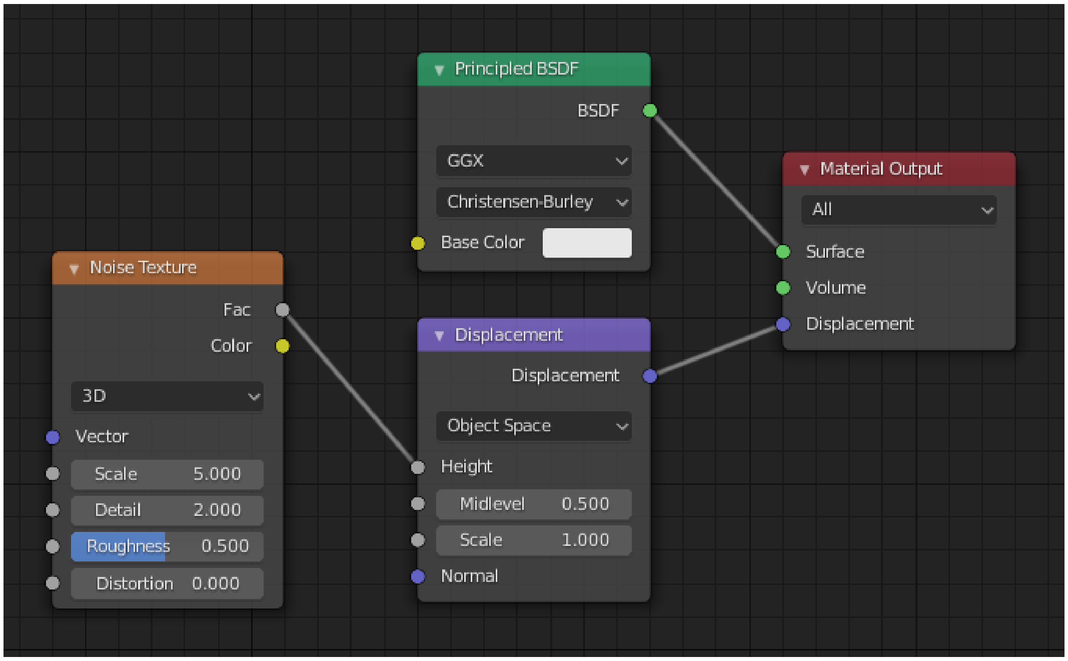
For rendering materials like skin or milk that require multiple scattering, subsurface scattering is more efficient and easier to control. Particularly the random walk method can accurately render such materials.

For materials such as clouds or smoke that do not have a well-defined surface, volume rendering is required. These look best with many scattering bounces, but in practice one might have to limit the number of bounces to keep render times acceptable.

Displacement

Detail can be added to the shape of a surface with displacement shaders.

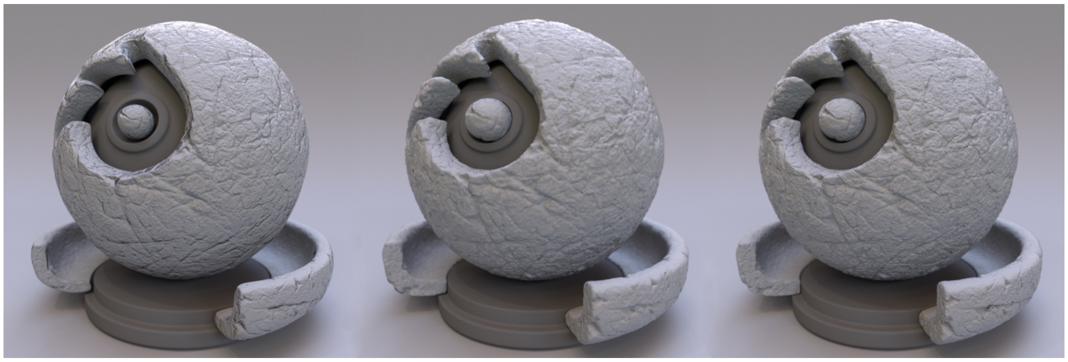
To create displacement, connect a Displacement or Vector Displacement node to the displacement input of the Material Output node. Procedural, painted or baked textures can then be connected to these nodes.



Principled BSDF
BSDF
GGX
Christensen-Burley
Base Color
Material Output
All
Surface
Volume
Displacement
Noise Texture
Fac
Color
3D
Vector
Scale 5.000
Detail 2.000
Roughness 0.500
Distortion 0.000
Displacement
Displacement
Object Space
Height
Midlevel 0.500
Scale 1.000
Normal

Typical displacement node setup.

Three displacement methods exist, with varying accuracy, performance and memory usage. The displacement method can be set per material in the Material Settings.



Bump only, displacement only, and displacement and bump combined.

Bump Only

The least accurate but most memory efficient method is bump mapping. This method does not actually alter the mesh surface, but merely changes the shading to make it seem so.

Bump maps are often used to add smaller details on a model, for example pores or wrinkles on skin.

For baked bump maps, 8-bit images are commonly used. However, 16 or 32-bit float maps can provide better looking results. When using image textures use Cubic interpolation to avoid stepping artifacts, these are more visible for bump maps than other types of textures.

Important

Because bump mapping is a fake effect, it can cause artifacts if the actual shape of the geometry is too different from the bump mapped shape. If this happens the strength of bump mapping should be reduced or actual displacement should be used.

Displacement Only

Cycles Only

The most accurate and memory intensive displacement method is to apply true displacement to the mesh surface.

It requires the mesh to be finely subdivided, which can be memory intensive. Adaptive Subdivision is the best way to subdivide the mesh, so that exactly the right amount of subdivision is used depending on the distance of the object to the camera.

For baked displacement maps, best results are achieved with 16 or 32-bit float maps, as 8-bit images often can not represent all the necessary detail.

See also

The Displace Modifier can also be used to displace a mesh.

Displacement and Bump

Both methods can be combined to use actual displacement for the bigger displacement and bump for the finer details. This can provide a good balance to reduce memory usage.

Once you subdivide the mesh very finely, it is better to use only actual displacement in Cycles. Keeping bump maps will then only increase memory usage and slow down renders.

Assignment

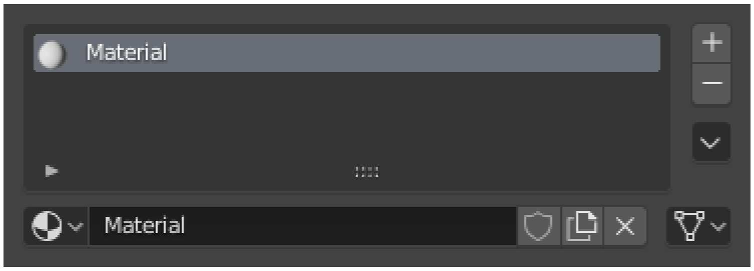
Reference

Panel:: *Material* ▸ *Material Slots*

Materials are data-blocks that can be created and then assigned to one or more objects. An object can also have multiple materials assigned in different material slots, which correspond to different parts of an object. If a smooth transition between materials is desired, then mixing shader nodes with a Mix shader is a better solution.

Material Slots

Material slots link materials to objects and meshes. By default objects only have a single material slot, which assigns a material to the entire object. If different parts of the mesh need different materials, multiple material slots can be created.



Material slots panel.

Slot List

The object's material slots and active material displayed in a List View.

Add Material Slot
: Add a new material slot on the object.

Remove Material Slot
    Remove a material slot from the object.

Copy Material
    Copy material shader nodes and settings to clipboard.

Paste Material
    Paste material shader nodes and settings from clipboard.

Copy Material to Selected
    Copy the same material assignment from the active to other selected objects.

Remove Unused Slots
    Removes all material slots not assigned to the object.

Data-Block

Material
    The Material Data-Block Menu for the selected material slot. Here new materials can be created, or existing materials can to the material slot.

Link
    Specifies whether the material is to be linked to the object or to the object data.

    The Link selector has two choices, Data and Object. These two menu choices determine whether the material is linked to the object or to the data (e.g. a mesh or curve). The Data menu item determines that this material will be linked to the mesh's data-block which is linked to the object's data-block. The Object menu item determines that the material will be linked to the object's data-block directly.

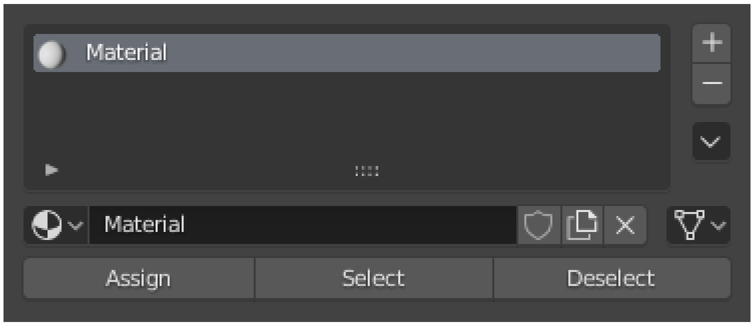
    This has consequences of course. For example, different objects may share the same mesh data-block. Since this data-block defines the shape of the object any change in Edit Mode will be reflected on all of those objects. Moreover, anything linked to that mesh data-block will be shared by every object that shares that mesh. So, if the material is linked to the mesh, every object will share it.

On the other hand, if the material is linked directly to the object data-block, the objects can have different materials and still share the same mesh.

Short explanation: If connected to the object, you can have several instances of the same Object Data using different materials. If linked to mesh data, you cannot. See [Data System](#) for more information.

Edit Mode

To assign materials to different parts of a mesh, enter Edit Mode on the mesh. Additional buttons will then appear in the material slots panel.



Material slots panel in Edit Mode.

Assign
: Assign active material slot and material to the selected faces in the mesh, strokes in a Grease Pencil, and similar for other object types.

Select
: Select faces assigned to the active material slot.

Deselect
: Deselect faces assigned to the active material slot.

Reusing Existing Materials

Blender is built to allow you to reuse *anything*, including material settings, between many objects. Instead of creating duplicate materials, you can simply reuse an existing material. There are several ways to do this using the Material's data-block menu:

Single Object – With the object selected, click the sphere located to the left of the Material name. A pop-up appears showing all the materials available in the current blend-file. To use one, just click on it.

Tip

Searching for Materials

The search field at the bottom of the material list allows you to search the names in the list. For example, by entering "wood" all existent materials are filtered so that only materials containing "wood" are displayed in the list.

Multiple Objects – In the 3D Viewport, with `Ctrl-L` you can quickly link all selected objects to the material (and other aspects) of the active object. Very useful if you need to set a large number of objects to the same material; just select all of them, then the object that has the desired material, and `Ctrl-L` links them to that "parent".

Deleting a Material

To delete a material, select the material and click X in the Available Materials List entry.

Although the material will seem to disappear immediately, the Delete action can depend on how the material is used elsewhere.

If the material is linked to the object and there are other objects which use this material, then the material will be removed from that object (but remain on all its other objects).

If the “Fake User” button has been lit in the Available Materials list, then the material will be retained when the file is saved, even if it has no users.

Only if it has 0 “real” users, and no “Fake” user, will the material be permanently deleted. Note that it will still remain in the Materials list until the blend-file is saved, but will have disappeared when the file is reloaded.

Multiple Materials

Normally, different colors or patterns on an object are achieved by adding textures to your materials. However, in some applications you can obtain multiple colors on an object by assigning different materials to the individual faces of the object.

To apply several materials to different faces of the same object, you use the Material Slots options in the Materials header panel.

The workflow for applying a second material to some faces of an object covered by a base material is as follows:

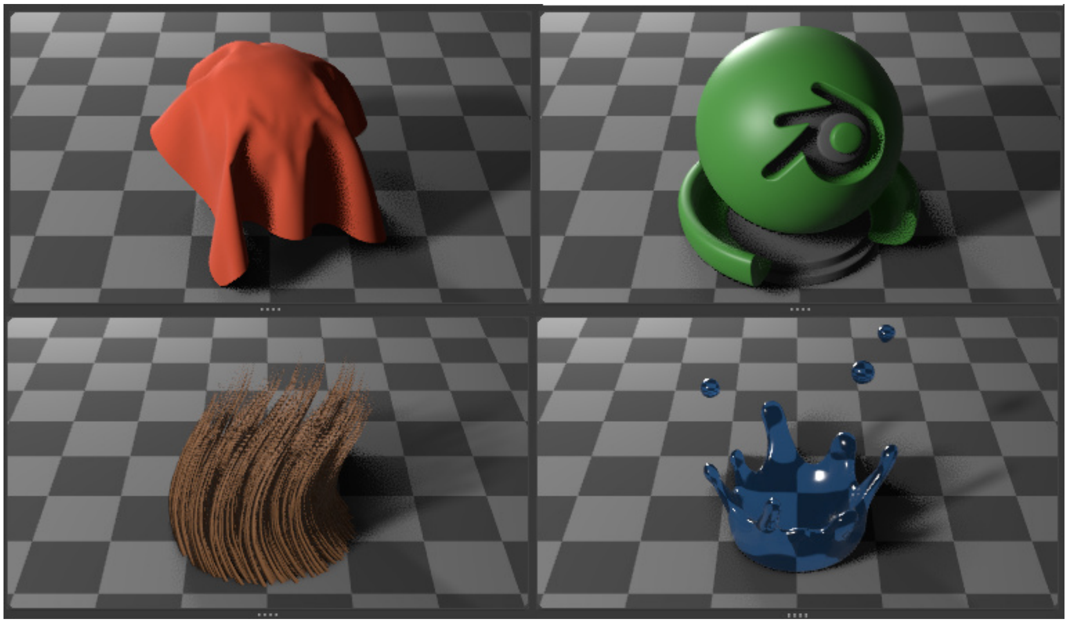
1. In Object Mode, create a base material.
2. Go into Edit Mode and Face Select (a new list will appear below the Active Material list with *Assign*, *Select*, *Deselect* buttons).
3. Select the faces to be colored with the second material.
4. In the Object Material Slots list, click the + button to create a new slot or select an existing material.
5. Click the *Assign* button, and the material will appear on the selected object faces.

Preview

The Preview panel gives a quick visualization of the active material applied in a simple scene.

Shape
: Preview the material on a Plane, Sphere, Cube, Hair, Shader Ball, Cloth or Fluid object. This shape is also used for previews when linking and appending materials.

Preview World *Cycles Only*
: Use the world from the current scene for lighting in the material preview.



Preview shapes.

Settings

Reference

Panel:: *Material ‣ Settings*

Renderer Settings

While shading nodes control the appearance, these settings control the quality and algorithms that each renderer uses to render the material.

- EEVEE specific settings
- Cycles specific settings

Pass Index

Pass Index

Index number for the *Material Index* render pass. This can be used to give a mask to a material and then be read with the ID Mask Node in the Compositor.

Note

Volume Objects are not supported.

Viewport Display

These settings control the 3D Viewport display in solid shading. They provide a faster alternative to full shader nodes, which may be too heavy or distracting for tasks like modeling, layout or sculpting.

Color

Diffuse or metal surface color.

Metallic

Blends between a non-metallic and metallic material model. A value of 1.0 gives a fully specular reflection tinted with the base color, without diffuse reflection or transmission. At 0.0 the material consists of a diffuse or transmissive base layer, with a specular reflection layer on top.

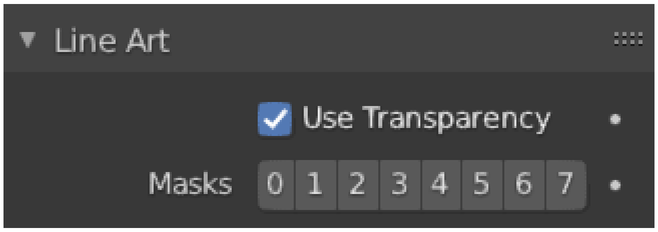
Roughness

Specifies microfacet roughness of the surface for metal and specular reflection.

Line Art

Reference

Panel:: *Material* ▸ *Line Art*



☑ Use Transparency

Line Art material properties.

Material Mask

Material masks are a way to provide Line Art extra information about faces that caused the occlusion. So edges occluded by those faces can be selected to have different styles.

Masks

The layer to include faces of the current material.

Levels

Faces with this material will behave as if it has set number of layers in occlusion.

Intersection Priority

Assigns an intersection priority value for this material. Note that this priority takes precedent over Object or Collection priority values. The intersection line will be included into the object with the higher intersection priority value.

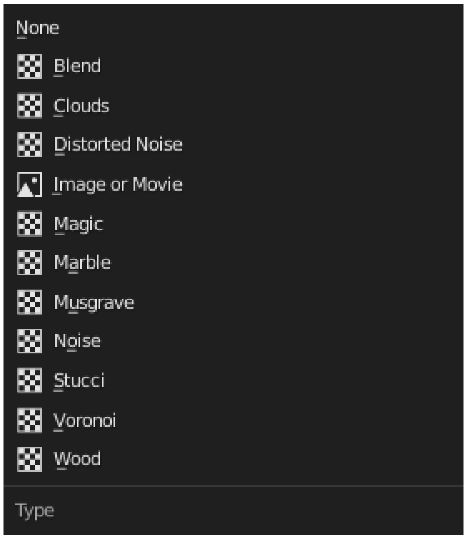
Legacy Textures

- Introduction
- Colors

Types

- Blend
- Clouds
- Distorted Noise
- Image or Movie
- Magic
- Marble
- Musgrave
- Noise
- Stucci
- Voronoi
- Wood

Introduction


None
Blend
Clouds
Distorted Noise
Image or Movie
Magic
Marble
Musgrave
Noise
Stucci
Voronoi
Wood
Type

The Texture Type list in the Texture panel of the Texture buttons.

Procedural textures are textures that are defined mathematically. They are generally relatively simple to use, because they do not need to be mapped in a special way. This does not mean that procedural textures cannot become very complex.

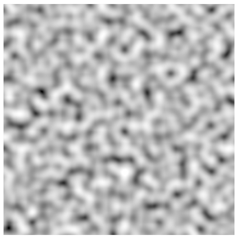
These types of textures are ‘real’ 3D. By that we mean that they fit together perfectly at the edges and continue to look like what they are meant to look like even when they are cut; as if a block of wood had really been cut in two. Procedural textures are not filtered or anti-aliased. This is hardly ever a problem: the user can easily keep the specified frequencies within acceptable limits.

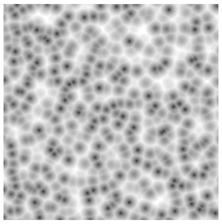
Common Options

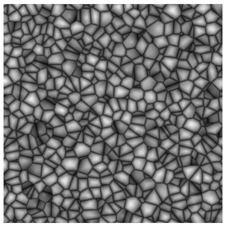
Noise Basis

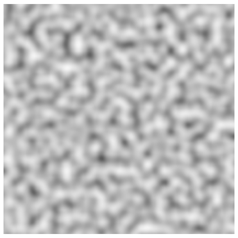
Each noise-based Blender texture (except Voronoi and Simple Noise) has a *Noise Basis* setting that allows the user to select which algorithm is used to generate the texture. This list includes the original Blender noise algorithm. The *Noise Basis* settings makes the procedural textures extremely flexible (especially *Musgrave*).

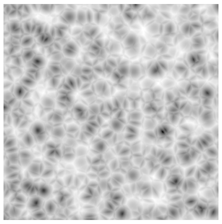
The *Noise Basis* governs the structural appearance of the texture:

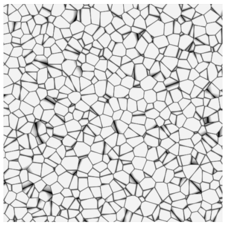

Blender Original.

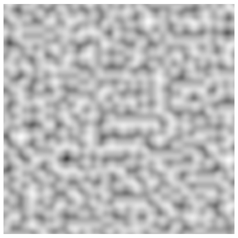

Voronoi F1.

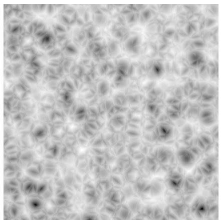

Voronoi F2-F1.

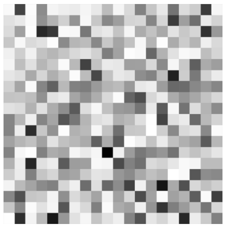

Original Perlin.

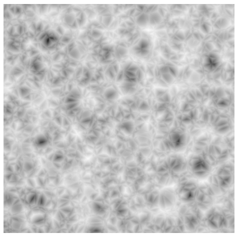

Voronoi F2.


Voronoi Crackle.


Improved Perlin.


Voronoi F3.


Cell Noise.



Voronoi F4.

There are two more possible settings for *Noise Basis*, which are relatively similar to *Blender Original*: Improved Perlin and Original Perlin.

Nabla

Almost all procedural textures in Blender use derivatives for calculating normals for texture mapping (except *Blend* and *Magic*). This is important for Normal and Displacement Maps. The strength of the effect is controlled with the *Nabla* number field.

Hints

Procedural textures can either produce colored textures, intensity only textures, textures with alpha values and normal textures. If intensity only ones are used the result is a black-and-white texture, which can be greatly enhanced by the use of ramps. If on the other hand you use ramps and need an intensity value, you have to switch on *No RGB* in the *Mapping* panel.

Colors

The color of a texture can be modified with the *Brightness*, *Contrast*, and *Saturation* buttons. All textures with RGB values, including *Images* and *Environment Maps*, may be modified with the RGB sliders.

Clamp

Set negative texture RGB and intensity values to zero, for some uses like displacement this option can be disabled to get the full range.

Multiply R, G, B

Tint the color of a texture by brightening each red, green and blue channel.

Brightness

Change the overall brightness/intensity of the texture.

Contrast

Change the contrast of the texture.

Saturation

Change the saturation of the texture.

Color Ramp

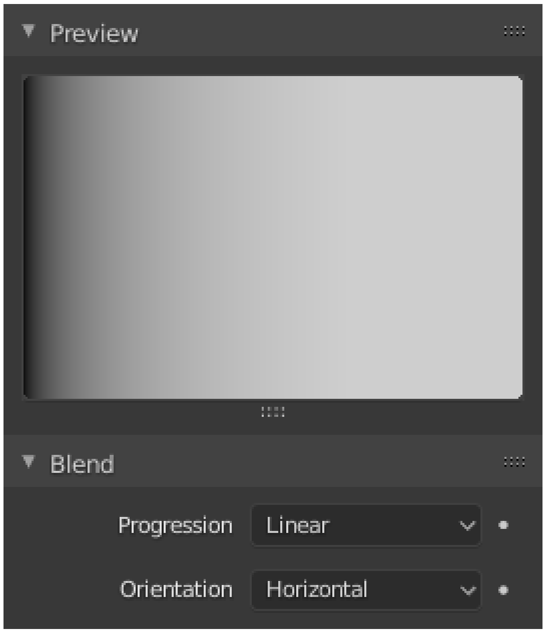
Activates a color ramp which allows you to remap the colors of a texture to new ones.

Blend

The Blend texture generates a smoothly interpolated progression. This is one of the most frequently used procedural textures. You can use blend textures to blend other textures together (with *Stencil*), or to create nice effects (especially with the *Mapping: Normal* trick).

Note

Remember that if you use a ramp to create a custom blending, you may have to use *No RGB*, if the *Mapping* value needs an intensity input.



Progression Linear
Orientation Horizontal

Blend Texture panels.

Options

Progression
    Profile of blend.

Linear
A linear progression.

Quadratic
A quadratic progression.

Easing
A flowing, nonlinear progression.

Diagonal
A diagonal progression.

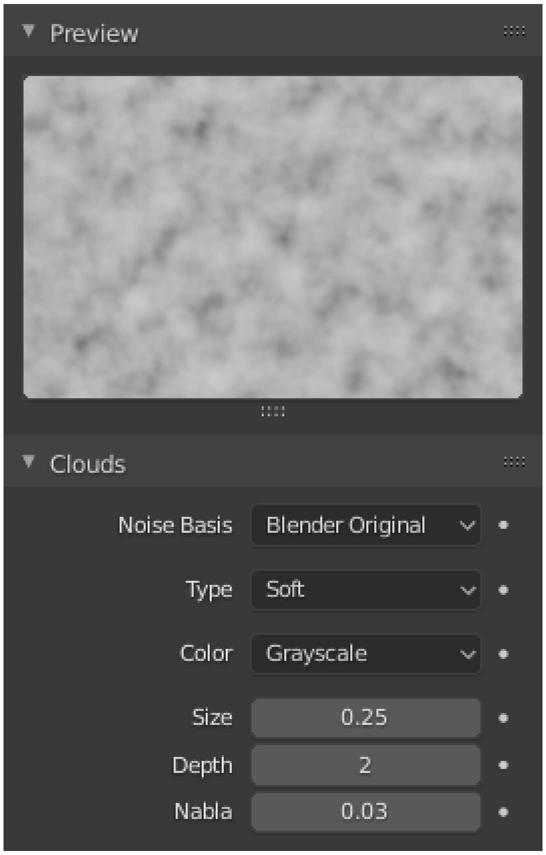
Spherical
A progression with the shape of a three-dimensional ball.

Quadratic Sphere
A quadratic progression with the shape of a three-dimensional ball.

Radial
A radial progression: *Horizontal* / *Vertical*. The direction of the progression is flipped a quarter turn.

Clouds

Clouds represent Perlin noise. In addition, each noise-based Blender texture (with the exception of Voronoi and simple noise) has a *Noise Basis* setting that allows the user to select which algorithm is used to generate the texture. This is often used for Clouds, Fire, Smoke. Well-suited to be used as a Bump map, giving an overall irregularity to the material.



Noise Basis: Blender Original
Type: Soft
Color: Grayscale
Size: 0.25
Depth: 2
Nabla: 0.03

Clouds Texture panels.

Options

Grayscale
: The standard noise, gives an intensity.

Color
: The noise gives an RGB value.

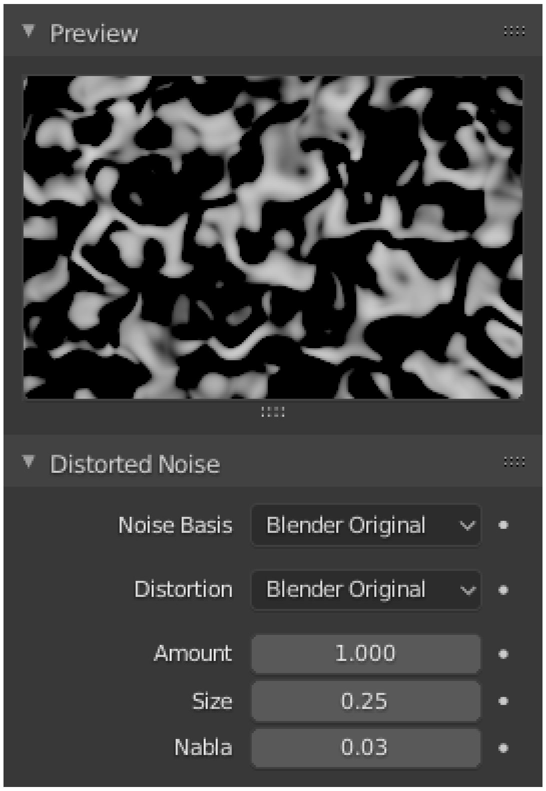
Noise
: *Soft* or *Hard*, changes contrast and sharpness.

Size
: The dimension of the Noise table.

Depth
: The depth of the *Clouds* calculation. A higher number results in a long calculation time, but also in finer details.

Distorted Noise

Distortion Noise takes the option that you pick from *Noise Basis* and filters it, to create hybrid pattern. It is often used for grunge but is also very complex and versatile.



Noise Basis: Blender Original

Distortion: Blender Original

Amount: 1.000

Size: 0.25

Nabla: 0.03

Distorted Noise Texture panels.

Options

Noise Basis
: The texture to be distorted.

Distortion
: The texture to use to distort another.

Amount

The amount that *Distortion Noise* affects *Basis*.

Size

The size of the noise generated.

Nabla

Almost all procedural textures in Blender use derivatives for calculating normals for texture mapping (except *Blend* and *Magic*). This is important for Normal and Displacement Maps. The strength of the effect is controlled with the *Nabla* number field.

Image or Movie

The term *Image Texture* simply means that a graphic image, which is a pixel grid composed of R, G, B, and sometimes Alpha values. It is used as the input source to the texture. As with other types of textures, this information can be used in a number of ways, not only as a simple “decal”.

Video textures are a some kind of Image textures and based on movie file or sequence of successive numbered separate images. They are added in the same way that image textures are.

When the Texture Type *Image or Movie* is selected, three new panels present themselves allowing to control most aspects of how image textures are applied: *Image*, *Image Sampling*, and *Image Mapping*.

About Image-Based Texturing

Texture images take up precious memory space, often being loaded into a special video memory bank that is very fast and very expensive, so it is often very small. So, keep the images as small as possible. A 64×64 image takes up only one fourth the memory of a 128×128 image.

For photorealistic rendering of objects in animations, often larger image textures are used, because the object might be zoomed in on in camera moves. In general, you want to use a texture sized proportionally to the number of pixels that it will occupy in the final render. Ultimately, you only have a certain amount of physical RAM to hold an image texture and the model and to provide workspace when rendering your image.

For the most efficient memory usage, image textures should be square, with dimensions as powers of 2, such as 32×32, 64×64, 128×128, 256×256, 1024×1024, 2048×2048, and 4096×4096.

If you can reuse images across different meshes, this greatly reduces memory requirements. You can reuse images if you map those areas of the meshes that “look alike” to a layout that uses the common image.

When using file textures, it is very important that you have Mapped the UVs of the mesh, and they are laid out appropriately.

You do not have to UV map the *entire* mesh. The sphere above on the left has some faces mapped, but other faces use procedural materials and textures. Only use UV textures for those portions of your mesh where you want very graphic, precise detail. For example, a model of a vase only needs UV texture for the rim where decorative artwork is incorporated. A throw pillow does not need a different image for the back as the front; in fact many throw pillows have a fabric (procedural material) back.

As another example, you should UV map both eyes of a head to the same image (unless you want one bloodshot and the other clear). Mapping both sides of a face to the same image might not be advisable, because the location of freckles and skin defects are not symmetrical. You could of course change the UV map for one side of the face to slightly offset, but it might be noticeable. Ears are another example where images or section of an image can be mapped to similar faces.

Options

Image
    The Image Data-Block Menu.

Alpha

Use the alpha channel information stored in the image. Where the alpha value in the image is less than 1.0, the object will be partially transparent and things behind it will be visible. Works with image formats that store transparency information.

Calculate
    Calculate an alpha based on the RGB values of the Image. Black (0, 0, 0) is transparent, white (1, 1, 1) opaque. Enable this option if the image

texture is a mask. Note that mask images can use shades of gray that result in semi-transparency, like ghosts, flames, and smoke/fog.

The image with various alpha and gray-scale values.

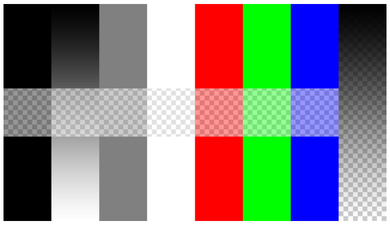


Image with *Use* alpha. The alpha values of the pixels are evaluated.

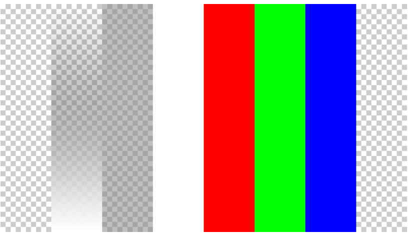
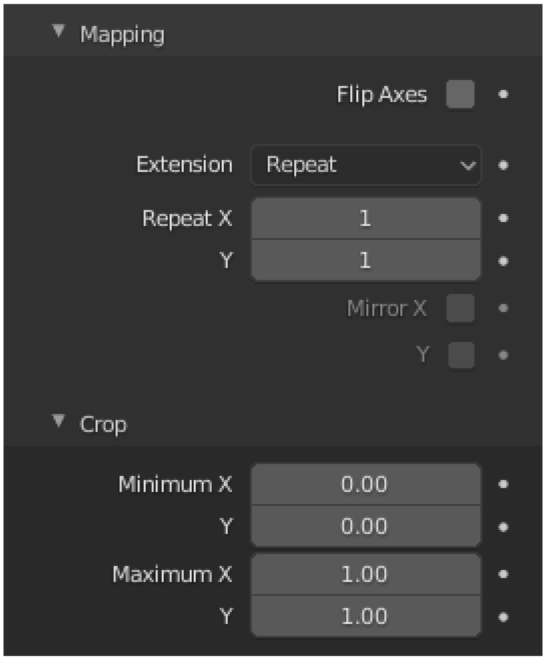


Image with *Calculate* alpha only, *Use Alpha* in the *Image* panel is disabled.

Invert

Reverses the alpha value. Use this option if the mask image has white where you want it transparent and vice versa.

Mapping



▼ Mapping

Flip Axes ☐

Extension: Repeat

Repeat X: 1

Y: 1

Mirror X ☐

Y ☐

▼ Crop

Minimum X: 0.00

Y: 0.00

Maximum X: 1.00

Y: 1.00

Image Mapping panel.

In the *Mapping* panel, you can control how the image is mapped or projected onto the 3D model.

Flip Axes
Rotates the image 90 degrees counterclockwise when rendered.

Extension
How the image is extrapolated beyond its original bounds.

Extend::	Outside the image the colors of the edges are extended.
Clip::	Clip to image size and set exterior pixels as transparent. Outside the image, an alpha value of 0.0 is returned. This allows you to 'paste' a small logo on a large object.
Clip Cube::	Clips to cubic-shaped area around the images and sets exterior pixels as transparent. The same as Clip, but now the 'Z' coordinate is calculated as well. An alpha value of 0.0 is returned outside a cube-shaped area around the image.
Repeat::	The image is repeated horizontally and vertically. Repeat X, Y X/Y repetition multiplier. Mirror X, Y Mirror on X/Y axes. These buttons allow you to map the texture as a mirror, or automatic flip of the image, in the corresponding X and/or Y direction.
Checker::	Checkerboards quickly made. You can use the option *size* on the *Mapping* panel as well to create the desired number of checkers. Tiles Even/Odd Set even/odd tiles. Distance

Governs the distance between the checkers in parts of the texture size.

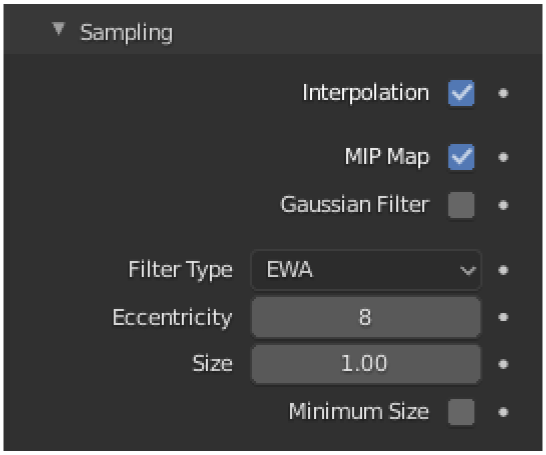
Crop

Minimum X, Y / Maximum X, Y

The offset and the size of the texture in relation to the texture space. Pixels outside this space are ignored. Use these to crop, or choose a portion of a larger image to use as the texture.

Sampling

In the *Sampling* panel you can control how the information is retrieved from the image.



Sampling

Interpolation ☑

MIP Map ☑

Gaussian Filter ☐

Filter Type: EWA

Eccentricity: 8

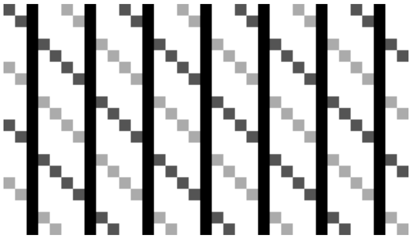
Size: 1.00

Minimum Size ☐

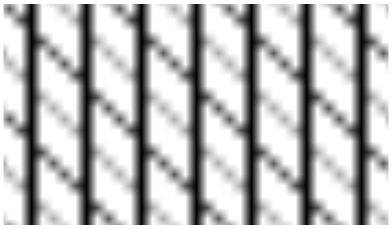
Image Sampling panel.

Interpolation

This option interpolates the pixels of an image. This becomes visible when you enlarge the picture. By default, this option is on. Turn this option off to keep the individual pixels visible and if they are correctly anti-aliased. This last feature is useful for regular patterns, such as lines and tiles; they remain ‘sharp’ even when enlarged considerably. Turn this image off if you are using digital photos to preserve crispness.



Enlarged Image texture without *Interpolation*.



Enlarged Image texture with *Interpolation*.

MIP Map

Mip-maps are precalculated, smaller, filtered textures for a certain size. A series of pictures is generated, each half the size of the former one. This optimizes the filtering process. By default, this option is enabled and speeds up rendering. When this option is off, you generally get a sharper image, but this can significantly increase calculation time if the filter dimension (see below) becomes large. Without mip-maps you may get varying pictures from slightly different camera angles, when the textures become very small. This would be noticeable in an animation.

Gaussian Filter

Used in conjunction with mip-mapping, it enables the mip-map to be made smaller based on color similarities. In game engines, you want your textures, especially your mip-map textures, to be as small as possible to increase rendering speed and frame rate.

Filter Type

Texture filter to use for image sampling. Just like a *pixel* represents a *pic* ture *el* ement, a *texel* represents a *tex* ture *el* ement. When a texture (2D texture space) is mapped onto a 3D model (3D model space), different algorithms can be used to compute a value for each pixel based on samples from several texels.

Box::	A fast and simple nearest-neighbor interpolation known as Monte Carlo integration.

EWA (Elliptical Weighted Average)::

One of the most efficient direct convolution algorithms developed by Paul Heckbert and Ned Greene in the 1980s. For each texel, EWA samples, weights, and accumulates texels within an elliptical footprint and then divides the result by the sum of the weights.

Eccentricity

Maximum Eccentricity. Higher values give less blur at distant/oblique angles, but is slower.

FELINE (Fast Elliptical Lines)::

Uses several isotropic probes at several points along a line in texture space to produce an anisotropic filter to reduce aliasing artifacts without considerably increasing rendering time.

Light Probes

Number of probes to use. An integer between 1 and 256. Further reading: McCormack, J; Farkas, KI; Perry, R; Jouppi, NP (1999) Simple and Table Feline: Fast Elliptical Lines for Anisotropic Texture Mapping, WRL

Area::

Area filter to use for image sampling.

Eccentricity

Maximum Eccentricity. Higher values give less blur at distant/oblique angles, but is slower.

Size

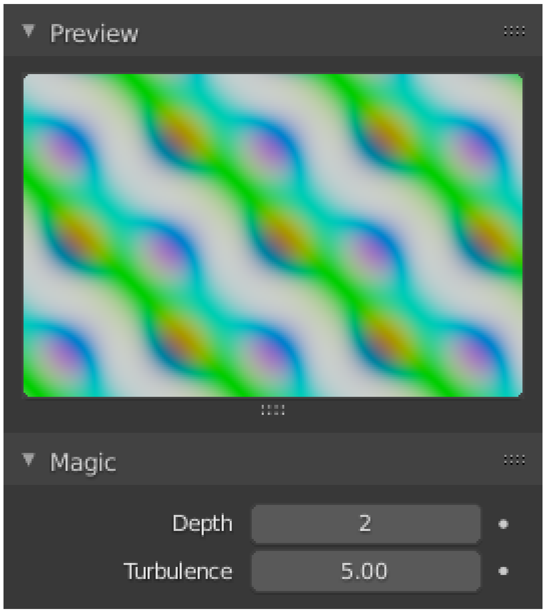
The filter size used in rendering, and also by the options *Mip Map* and *Interpolation*. If you notice gray lines or outlines around the textured object, particularly where the image is transparent, turn this value down from 1.0 to 0.1 or so.

Minimum Size

Use Filter Size as a minimal filter value in pixels.

Magic

The Magic Texture node is used to add a psychedelic color texture. It can be used for “Thin Film Interference” if you set *Mapping* to *Reflection* and use a relatively high *Turbulence*. The RGB components are generated independently with a sine formula.



Depth: 2

Turbulence: 5.00

Magic Texture panels.

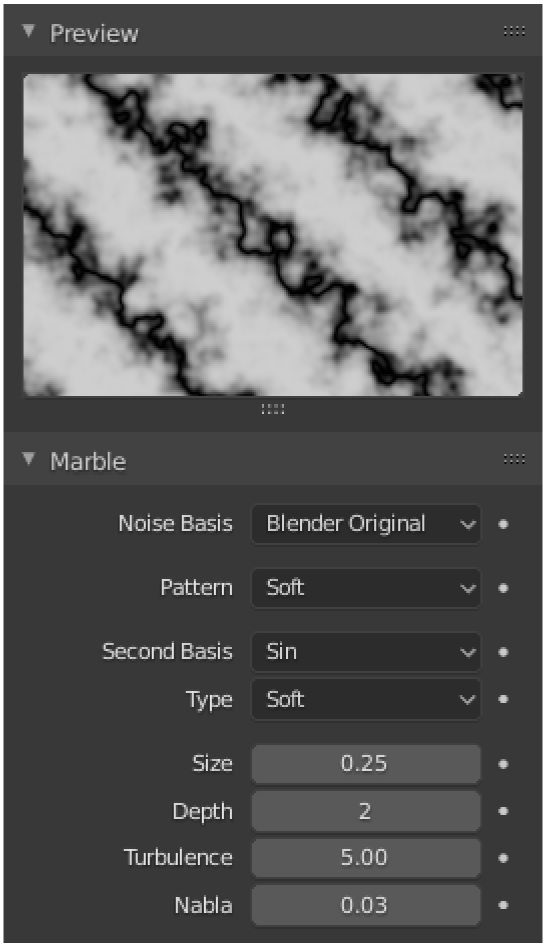
Options

Depth
: The depth of the calculation. A higher number results in a long calculation time, but also in finer details.

Turbulence
: The strength of the pattern.

Marble

The marble texture is used to generate marble, fire, or noise with a structure. Bands are generated based on the sine, saw, or triangular formula and noise turbulence.



Noise Basis: Blender Original
Pattern: Soft
Second Basis: Sin
Type: Soft
Size: 0.25
Depth: 2
Turbulence: 5.00
Nabla: 0.03

Marble Texture panels.

Options

Marble Type
  Three settings for soft to more clearly defined *Marble*.

Soft, Sharp, Sharper

Noise basis

Shape of wave to produce bands.

Sine, Saw, Triangle

Noise Type

The noise function works with two methods.

Soft, Hard

Size

The dimensions of the noise table.

Depth

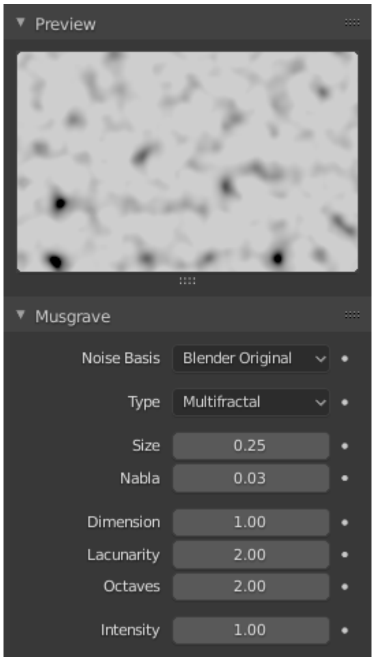
The depth of the *Marble* calculation. A higher value results in greater calculation time, but also in finer details.

Turbulence

The turbulence of the sine bands.

Musgrave

The musgrave texture is used to generate organic materials, but it is very flexible. You can do nearly everything with it.


Preview

Musgrave

Noise Basis: Blender Original
Type: Multifractal
Size: 0.25
Nabla: 0.03
Dimension: 1.00
Lacunarity: 2.00
Octaves: 2.00
Intensity: 1.00

Musgrave Texture panels.

Options

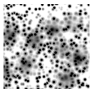
Type

This procedural texture has five noise types on which the resulting pattern can be based and they are selectable from a select menu at the top of the tab. The five types are:

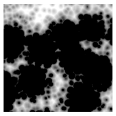
- Hetero Terrain
- Fractal Brownian Motion (fBm)
- Hybrid Multifractal
- Ridged Multifractal
- Multifractal

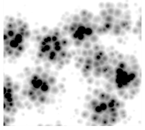
These noise types determine the manner in which Blender layers successive copies of the same pattern on top of each other at varying contrasts and scales.

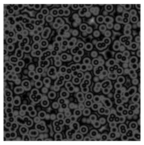
Examples with Basis: Voronoi: F1, Dimension: 0.5, Lacunarity: 0.15, Octave: 2.0.

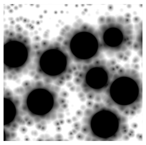


Hetero Terrain.









Fractal Brownian Motion.	Hybrid Multifractal.	Ridged Multifractal.	Multifractal.

The main noise types have four characteristics:

Dimension

Fractal dimension controls the contrast of a layer relative to the previous layer in the texture. The higher the fractal dimension, the higher the contrast between each layer, and thus the more detail shows in the texture.

Lacunarity

Lacunarity controls the scaling of each layer of the Musgrave texture, meaning that each additional layer will have a scale that is the inverse of the value which shows on the button. i.e. Lacunarity = 2 –> Scale = 1/2 original.

Octaves

Octave controls the number of times the original noise pattern is overlaid on itself and scaled/contrasted with the fractal dimension and lacunarity settings.

Intensity

Light intensity. Called *Offset* for *Hetero Terrain*.

The *Hybrid Multifractal* and *Ridged Multifractal* types have these additional settings:

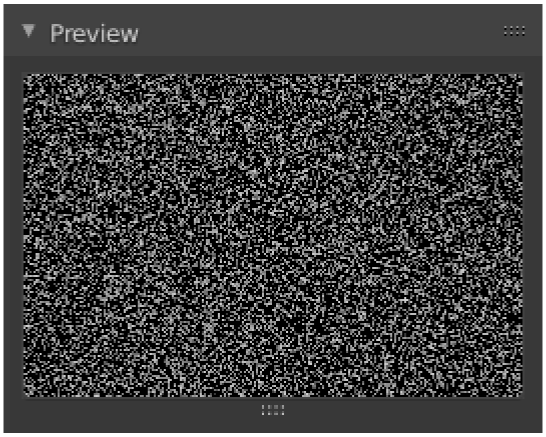
Offset

Both have a “Fractal Offset” button that serves as a “sea level” adjustment and indicates the base height of the resulting bump map. Bump values below this threshold will be returned as zero.

Gain

Setting which determines the range of values created by the function. The higher the number, the greater the range. This is a fast way to bring out additional details in a texture where extremes are normally clipped off.

Noise



Noise Texture panel.

Although this looks great, it is not Perlin Noise! This is a true, randomly generated Noise. This gives a different result every time, for every frame, for every pixel.

Options

There are no options for this noise.

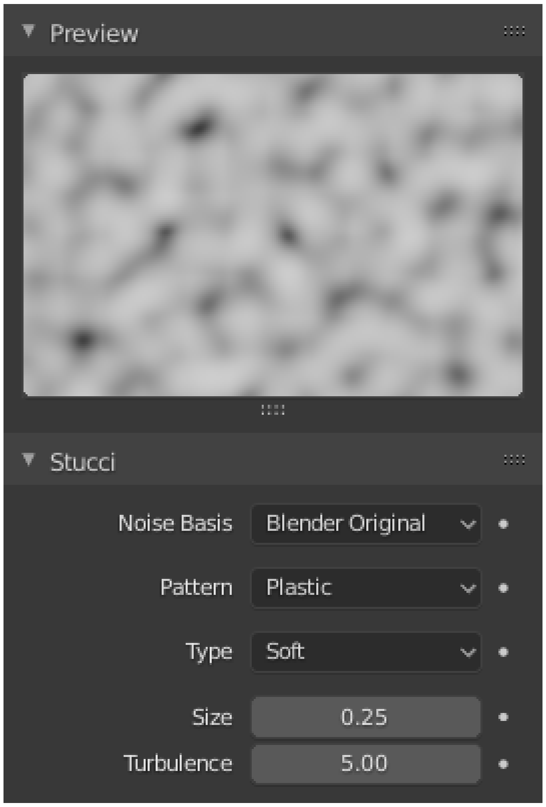
Often used for

White noise in an animation. This is not well suited if you do not want an animation. For material displacement or bump, use clouds instead.

Result(s)

Intensity.

Stucci



Noise Basis: Blender Original
Pattern: Plastic
Type: Soft
Size: 0.25
Turbulence: 5.00

Stucci Texture panels.

The *Stucci* texture is based on noise functions. It is often used for stone, asphalt, or oranges, normally for bump mapping to create grainy surfaces.

Options

Plastic / Wall In / Wall out
: Plastic is the standard Stucci, while the “walls” is where Stucci gets it name. This is a typical wall structure with holes or bumps.

Soft / Hard
: There are two methods available for working with Noise.

Size

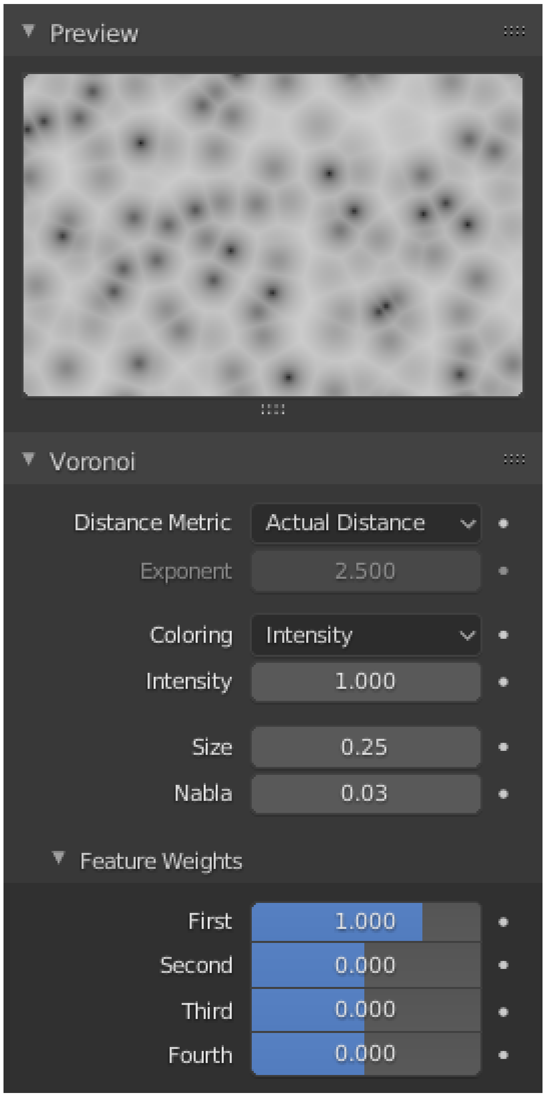
Dimension of the Noise table.

Turbulence

Depth of the *Stucci* calculations.

Voronoi

The Voronoi texture is used to generate very convincing Metal, especially the “Hammered” effect. Organic shaders (e.g. scales, veins in skin).



Distance Metric	Actual Distance
Exponent	2.500
Coloring	Intensity
Intensity	1.000
Size	0.25
Nabla	0.03
First	1.000
Second	0.000
Third	0.000
Fourth	0.000

Voronoi Texture panels.

Options

Distance Metric

This procedural texture has seven Distance Metric options. These determine the algorithm to find the distance between cells of the texture. These options are:

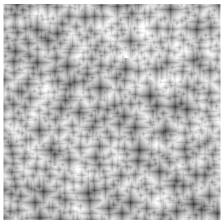
- Minkowski
- Minkowski 4
- Minkowski 1/2
- Chebychev
- Manhattan
- Distance Squared
- Actual Distance

The *Minkowski* setting has a user definable value (the *Exponent* button) which determines the Minkowski exponent e of the distance function:

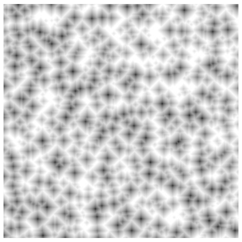
$$(x^{e} + y^{e} + z^{e})^{1/e}$$

A value of one produces the *Manhattan* distance metric, a value less than one produces stars (at 0.5, it gives a *Minkowski 1/2*), and higher values produce square cells (at 4.0, it gives a *Minkowski 4*, at 10.0, a *Chebychev*). So nearly all Distance Settings are basically the same – a variation of *Minkowski*.

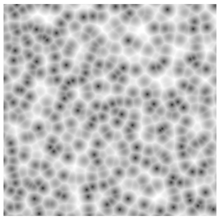
You can get irregularly-shaped rounded cells with the *Actual Distance / Distance Squared* options.



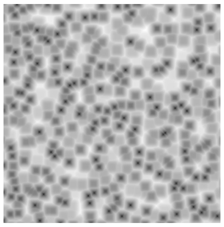
Minkowski Exponent: 0.5 (Minkowski 1/2).



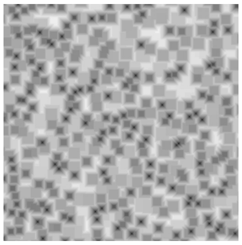
Minkowski Exponent: 1 (Manhattan).



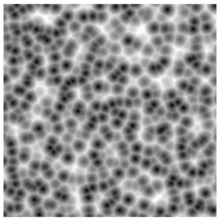
Minkowski Exponent: 2 (Actual Distance).



Minkowski Exponent: 4 (Minkowski 4).



Minkowski Exponent: 10 (Chebychev).



Distance Squared (more contrast than Actual Distance).

Feature Weights

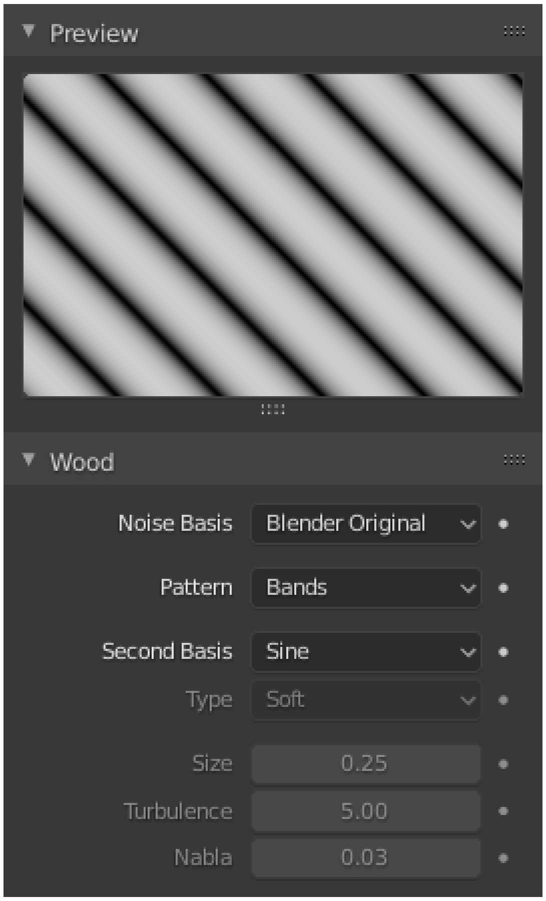
These four sliders at the bottom of the Voronoi panel represent the values of the four Worley constants, which are used to calculate the distances between each cell in the texture based on the distance metric. Adjusting these values can have some interesting effects on the end result…

Coloring

Four settings (*Intensity*, *Position*, *Position and Outline*, and *Position, Outline, and Intensity*) that can use four different noise basis as methods to calculate color and intensity of the texture output. This gives the Voronoi texture you create with the "Worley Sliders" a completely different appearance and is the equivalent of the noise basis setting found on the other textures.

Wood

The wood texture is used to generate wood and ring-shaped patterns.



Noise Basis Blender Original
Pattern Bands
Second Basis Sine
Type Soft
Size 0.25
Turbulence 5.00
Nabla 0.03

Wood Texture panels.

Options

Noise Basis
    Shape of wave to produce bands.

    Sine, Saw, Triangle

Wood Type

Set the bands to either straight or ring-shaped, with or without turbulence.

Bands, Rings, Band Noise, Ring Noise

Noise Type

There are two methods available for the Noise function.

Soft, Hard

Size

Dimension of the Noise table.

Turbulence

Turbulence of the Band Noise and Ring Noise types.

Shader Nodes

- Introduction
    - Shaders
    - Textures
    - More
    - Open Shading Language
- Input
    - Ambient Occlusion
    - Attribute Node
    - Bevel Node
    - Camera Data Node
    - Fresnel Node
    - Geometry Node
    - Curves Info Node
    - Layer Weight Node
    - Light Path Node
    - Object Info Node
    - Particle Info Node
    - Point Info
    - RGB Node
    - Tangent Node
    - Texture Coordinate Node
    - UV Map Node
    - Value Node
    - Color Attribute Node
    - Volume Info Node
    - Wireframe Node
- Output
    - AOV Output Node
    - Material Output Node
    - Light Output Node
    - World Output Node
- Shader
    - Add Shader

  - Background
  - Diffuse BSDF
  - Emission
  - Glass BSDF
  - Glossy BSDF
  - Hair BSDF
  - Holdout
  - Mix Shader
  - Metallic BSDF
  - Principled BSDF
  - Principled Hair BSDF
  - Principled Volume
  - Ray Portal BSDF
  - Refraction BSDF
  - Specular BSDF
  - Subsurface Scattering
  - Toon BSDF
  - Translucent BSDF
  - Transparent BSDF
  - Sheen BSDF
  - Volume Absorption
  - Volume Scatter
- Texture
  - Brick Texture Node
  - Checker Texture Node
  - Environment Texture Node
  - Gabor Texture Node
  - Gradient Texture Node
  - IES Texture Node
  - Image Texture Node
  - Magic Texture Node
  - Musgrave Texture Node
  - Noise Texture Node
  - Point Density Node
  - Sky Texture Node
  - Voronoi Texture Node
  - Wave Texture Node

  - White Noise Texture Node
- Color
  - Brightness/Contrast Node
  - Gamma Node
  - Hue/Saturation/Value Node
  - Invert Color Node
  - Light Falloff Node
  - Mix Color Node
  - RGB Curves Node
- Vector
  - Bump Node
  - Displacement Node
  - Mapping Node
  - Normal Node
  - Normal Map Node
  - Vector Curves Node
  - Vector Displacement Node
  - Vector Rotate Node
  - Vector Transform Node
- Converter
  - Blackbody Node
  - Clamp Node
  - Color Ramp Node
  - Combine Color Node
  - Combine XYZ Node
  - Float Curve
  - Map Range Node
  - Math Node
  - Mix Node
  - RGB to BW Node
  - Separate Color Node
  - Separate XYZ Node
  - Shader To RGB
  - Vector Math Node
  - Wavelength Node
- Group
  - Group Input

    - Group Output
    - Node Groups
- Open Shading Language
    - Script Node
    - Writing Shaders
    - Closures
    - Attributes
    - Trace
    - Metadata

Introduction

Materials, lights and backgrounds are all defined using a network of shading nodes. These nodes output values, vectors, colors and shaders.

Shaders

An important concept to understand when building node setups is that of the *shader socket*. The output of all surface and volume shaders is a shader, describing lighting interaction at the surface or of the volume, rather than the color of the surface.

There are a few types of shaders available as nodes:

BSDF shader
Describe light reflection, refraction and absorption at an object surface.

Emission shader
Describe light emission at an object surface or in a volume.

Volume shader
Describe light scattering inside a volume.

Background shader
Describe light emission from the environment.

Each shader node has a color input, and outputs a shader. These can then be mixed and added together using Mix and Add Shader nodes. No other operations are permitted. The resulting output can then be used by the renderer to compute all light interactions, for direct lighting or global illumination.

See also

Shaders

Textures

Blender has various built in procedural texture nodes, with texture coordinates and various parameters as input, and a color or value as output. No texture data-blocks are needed; instead node groups can be used for reusing texture setups.

For UV mapping and texture painting in the 3D Viewport, the Image Texture node must be used. When setting such a node as active, it will be displayed in the 3D Viewport while using Texture color mode. This method can be used to preview painted textures while texture painting.

The default texture coordinates for all nodes are Generated coordinates, except for Image textures that use UV coordinates by default. Each node includes some options to modify the texture mapping and resulting color, and these can be edited in the texture properties.

See also

Textures

More

Nodes for geometric data, texture coordinates, layering shaders and non-physically-based tricks can be found in:

- Vector Nodes
- Color Nodes
- Converter Nodes

Open Shading Language

In Cycles, custom nodes can be written using the Open Shading Language.

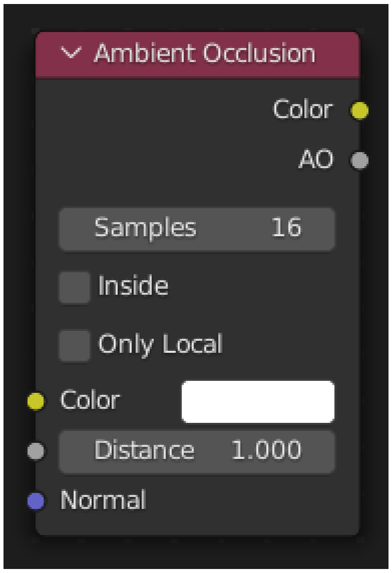
See also

Open Shading Language

Input

- Ambient Occlusion
- Attribute Node
- Bevel Node
- Camera Data Node
- Fresnel Node
- Geometry Node
- Curves Info Node
- Layer Weight Node
- Light Path Node
- Object Info Node
- Particle Info Node
- Point Info
- RGB Node
- Tangent Node
- Texture Coordinate Node
- UV Map Node
- Value Node
- Color Attribute Node
- Volume Info Node
- Wireframe Node

Ambient Occlusion



Samples 16

☐ Inside

☐ Only Local

Distance 1.000

The *Ambient Occlusion* shader computes how much the hemisphere above the shading point is occluded. This can be used for procedural texturing, for example to add weathering effects to corners only.

For Cycles, this is an expensive shader and can slow down render significantly. If render time is a concern, using Pointiness from the Geometry node or baking Ambient Occlusion will result in faster renders.

Note

Cycles Only

The Ambient Occlusion node will not produce a valid result when:

- The object is either a Caustic caster or Caustic receiver while the scene contains an active Caustic caster, Caustic receiver, and Shadow Caustic Light.
- Open Shading Language is active while using the OptiX rendering backend.

Inputs

Color

Tint for AO output color.

Distance

Distance up to which other objects are considered to occlude the shading point.

Normal

Normal used for ambient occlusion; if nothing is connected the default shading normal is used.

Properties

Samples
Number of samples to use for ray-traced ambient occlusion sampling. Keep as low as possible for optimal performance.

Inside
Detect convex rather than concave shapes, by computing occlusion inside mesh.

Only Local *Cycles Only*
Only detect occlusion from the object itself, and not others.

Outputs

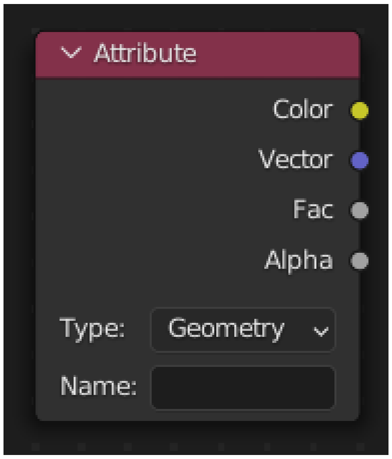
Color
Ambient occlusion with color tint.

AO
Ambient occlusion factor without color tint.

Example

White AO shader.

Attribute Node



The *Attribute* node allows you to retrieve attributes attached to an object or mesh.

Inputs

This node has no inputs.

Properties

Name
: Name of the attribute.

Type
: Specifies the type of the attribute.

Geometry::	The attribute is associated with the geometry of the object, and its value varies from vertex to vertex, or within the volume of the object. Most geometry attributes are directly accessible through the various input nodes, except for these: Ocean Foam Gives a scalar defining where foam might appear when using an Ocean Modifier. This depends on the name you give this property.

See also

For a full list of options see This Thread on the Blender Stack Exchange.

Object::

The attribute name specifies a custom property name, or an RNA path to a built-in property (like the single property driver variables).
The values of attributes of this type are defined once per object. The name or path is looked up first in the object data-block, followed by the mesh data-block if not found. Custom properties have priority over built-in ones.
The property value must be an integer, float, boolean, or a vector of 1 to 4 floats or ints; properties of other types are ignored. If a suitable property is not found, all sockets of the node, including *Alpha*, output 0.

Tip

The `color` attribute will output the value of the Color field in the Viewport Display panel of the object, unless overridden by a custom property.

Instancer::

Similar to *Object*, but the attribute is looked up in the instancer particle system settings, followed by Geometry Node instance attributes (searching from the innermost instancing layer to outer ones), and finally in the instancer object. If the current object is not instanced, or the property is not found, it falls back to the *Object* mode.

Warning

Currently only up to 4 layers of Geometry Node instancing are searched.

View Layer::

The attribute is looked up in the current View Layer, Scene and World, using the same lookup logic as *Object*, and likewise producing all zero outputs including *Alpha* if not found. Attributes of this type have the same uniform value throughout the whole Render Layer.

Tip

This gives access to a number of useful built-in properties, for example:

`color` **or** **`world.color`**
: Outputs the value of the Color field in the Viewport Display panel of the World properties.

`render.resolution_x`, **`render.resolution_y`**
: Outputs the current rendering resolution.

`camera.data.angle_x`, **`camera.data.angle_y`**,
: Outputs the effective field of view of the active Camera.

See also

An alternative method to access the same set of properties is to use driver Context Properties, possibly with a manually emulated lookup fallback chain.

Outputs

Color
: RGB color interpolated from the attribute.

Vector
: XYZ vector interpolated from the attribute.

Factor
: Scalar value interpolated from the attribute.

Alpha

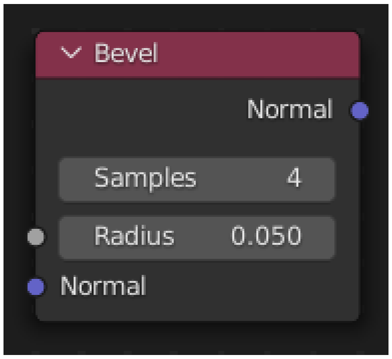
Alpha channel of the attribute, when available. If the attribute has no alpha channel, generally defaults to 1.

Warning

Currently, only *View Layer* attributes are supported in shaders used for the World or Light Objects.

Bevel Node

Cycles Only



Samples 4

Radius 0.050

The Bevel shader node can be used for rendering rounded corners. Like bump mapping, this does not modify the actual geometry, only the shading is affected. Slight rounding on edges helps to capture specular highlights that you would also see in the real world.

Note that this is a very expensive shader, and may slow down renders by 20% even if there is a lot of other complexity in the scene. For that reason, we suggest to mainly use this for baking or still frame renders where render time is not as much of an issue. The Bevel Modifier is a faster option when it works, but sometimes fails on complex or messy geometry.

Note

The Bevel node will not produce a valid result when:

- **The object is either a Caustic caster** or Caustic receiver while the scene contains an active Caustic caster, Caustic receiver, and Shadow Caustic Light.
- Open Shading Language is active while using the OptiX rendering backend.

Inputs

Radius

Width of the bevel effect on edges.

Normal

Normal to apply bevel on top of, to be combined with a Bump Node for example. When not connected, uses the surface normal.

Properties

Samples

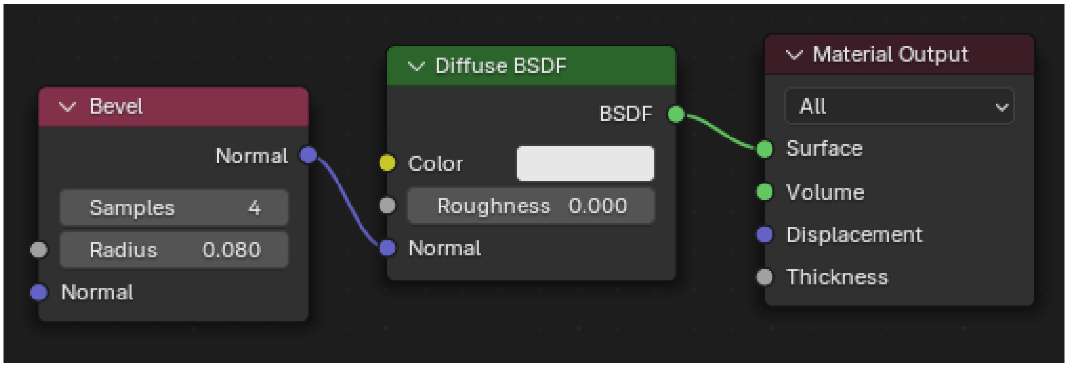
Number of samples to take for each shader evaluation. More samples give more accurate results, but are also slower to render. The default value of 4 works well for most cases, with any noise resolved by using more AA samples.

Outputs

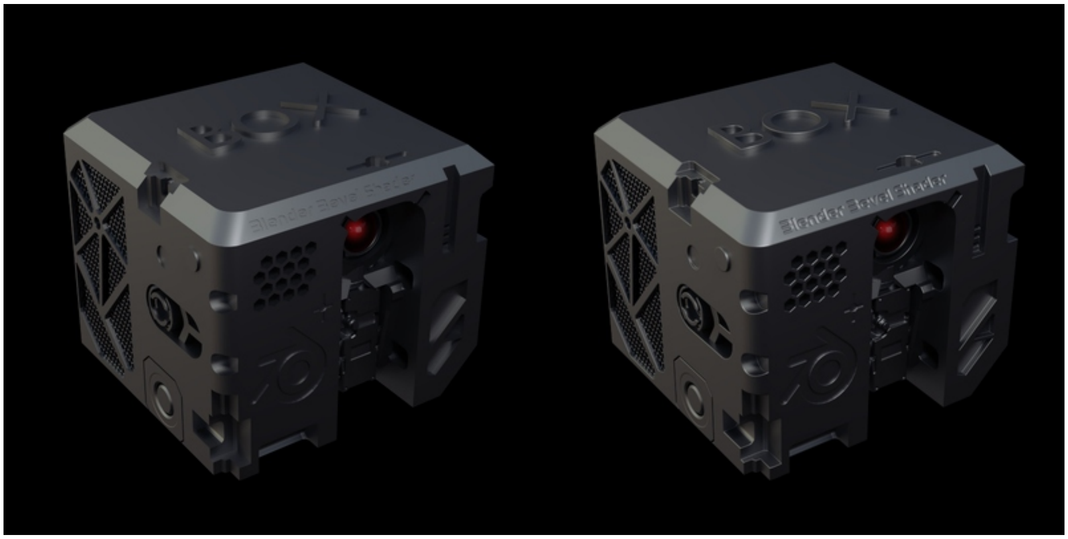
Normal

Standard normal output.

Examples

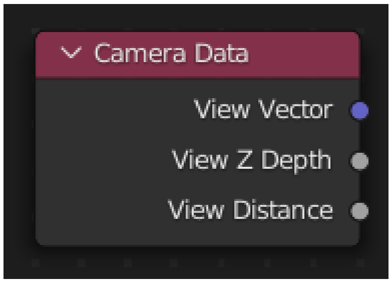

Bevel
Normal
Samples 4
Radius 0.080
Normal
Diffuse BSDF
BSDF
Color
Roughness 0.000
Normal
Material Output
All
Surface
Volume
Displacement
Thickness

A minimal node setup for working with the Bevel node.



Bevel shader bringing out specular highlights on the edges.

Camera Data Node



The *Camera Data* node returns information about the shading point relative to the camera. This could be used for example to change the shading of objects further away from the camera, or make custom fog effects.

Inputs

This node has no inputs.

Properties

This node has no properties.

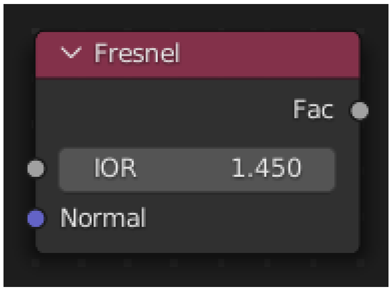
Outputs

View Vector
: A normalized vector, in camera space, from the camera to the shading point.

View Z Depth
: The distance each pixel is away from the camera.

View Distance
: Distance from the camera to the shading point.

Fresnel Node



The *Fresnel* or *Dielectric Fresnel* node computes how much light is reflected off a layer, where the rest will be refracted through the layer. The resulting weight can be used for layering shaders with the *Mix Shader* node. It is dependent on the angle between the surface normal and the viewing direction.

The most common use is to mix between two BSDFs using it as a blending factor in a Mix Shader node. For a simple glass material you would mix between a glossy refraction and glossy reflection. At grazing angles more light will be reflected than refracted as happens in reality.

For a two-layered material with a diffuse base and a glossy coating, you can use the same setup, mixing between a diffuse and glossy BSDF. By using the Fresnel as the blending factor you are specifying that any light which is refracted through the glossy coating layer would hit the diffuse base and be reflected off that.

Inputs

IOR
: Index of refraction (IOR) of the material being entered.

Normal
: Input meant for plugging in bump or normal maps which will affect the output.

Properties

This node has no properties.

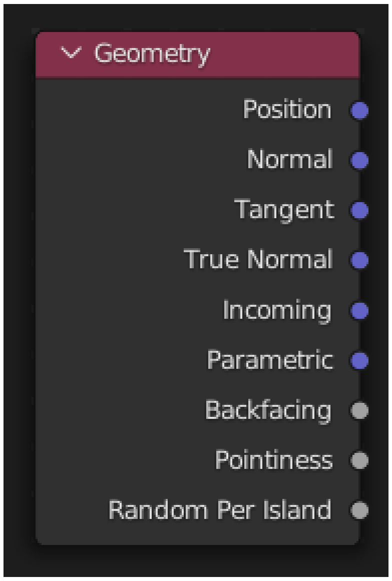
Outputs

Factor

Fresnel weight, indicating the probability with which light will reflect off the layer rather than passing through.

Geometry Node

The *Geometry* node gives geometric information about the current shading point. All vector coordinates are in *World Space*. For volume shaders, only the position and incoming vector are available.


Geometry
Position
Normal
Tangent
True Normal
Incoming
Parametric
Backfacing
Pointiness
Random Per Island

Inputs

This node has no inputs.

Properties

This node has no properties.

Outputs

Position
: Position of the shading point.

Normal
: Shading normal at the surface (includes smooth normals and bump mapping).

Tangent
: Tangent at the surface.

True Normal
: Geometry or flat normal of the surface.

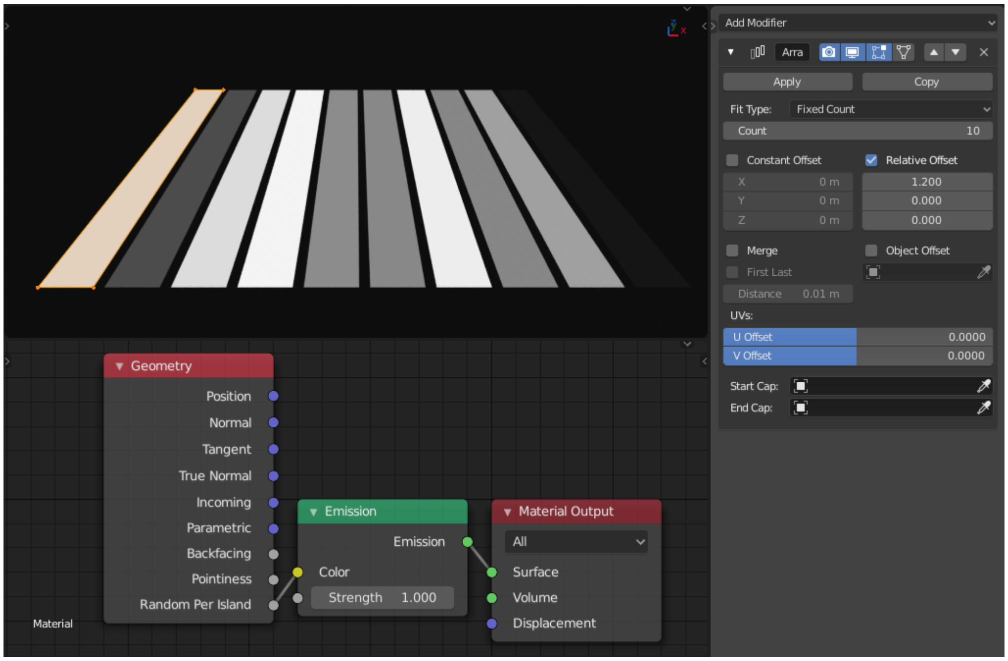
Incoming
: Vector pointing towards the point the shading point is being viewed from.

Parametric
: Parametric coordinates of the shading point on the surface. To area lights it outputs its UV coordinates in planar mapping and in spherical coordinates to point lights.

Backfacing
: 1.0 if the face is being viewed from the back side, 0.0 for the front side.

Pointiness *Cycles Only*
: An approximation of the curvature of the mesh per vertex. Lighter values indicate convex angles, darker values indicate concave angles. It allows you to do effects like dirt maps and wear-off effects.

Random per Island *Cycles Only*
: A random value for each connected component (island) of the mesh. It is useful to add variations to meshes composed of separated units like tree leaves, wood planks, or curves of multiple splines.



Geometry
Position
Normal
Tangent
True Normal
Incoming
Parametric
Backfacing
Pointiness
Random Per Island
Emission
Emission
Color
Material Output
Surface
Volume
Displacement
Material
Add Modifier
Arra
Apply
Copy
UVs:

Strength 1.000

All

Fit Type: Fixed Count

Count 10

☐ Constant Offset

X 0 m

Y 0 m

Z 0 m

☑ Relative Offset

1.200

0.000

0.000

☐ Merge

☐ First Last

Distance 0.01 m

☐ Object Offset

U Offset 0.0000

V Offset 0.0000

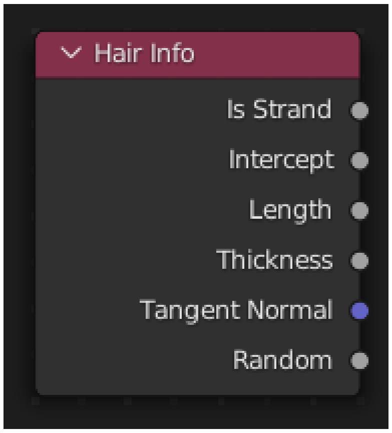
Start Cap:

End Cap:

Get a random value for each instance of the mesh when using an Array modifier.

Curves Info Node

The *Curves Info* node gives access to Hair information.


Hair Info
Is Strand
Intercept
Length
Thickness
Tangent Normal
Random

Inputs

This node has no inputs.

Properties

This node has no properties.

Outputs

Is Strand

Outputs 1 when the shader is acting on a strand, otherwise 0.

Intercept

The point along the strand where the ray hits the strand (1 at the tip and 0 at the root).

Length

The total measurement from the root to the tip of the strand, interpreted as a grayscale value from 0 to infinity.

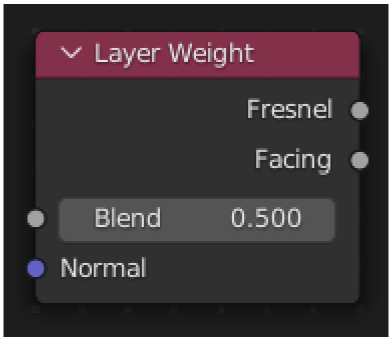
Thickness
    The thickness of the strand at the point where the ray hits the strand.

Tangent Normal
    Tangent normal of the strand.

Random
    A random per-curve value in the range from 0 to 1. It can for example be used in combination with a color ramp, to randomize the curve’s color.

Layer Weight Node

The *Layer Weight* node outputs a weight typically used for layering shaders with the *Mix Shader* node.



Blend 0.500

Inputs

Blend
: Bias the output towards all 0 or all 1. Useful for uneven mixing of shaders.

Normal
: Input meant for plugging in bump or normal maps which will affect the output.

Properties

This node has no properties.

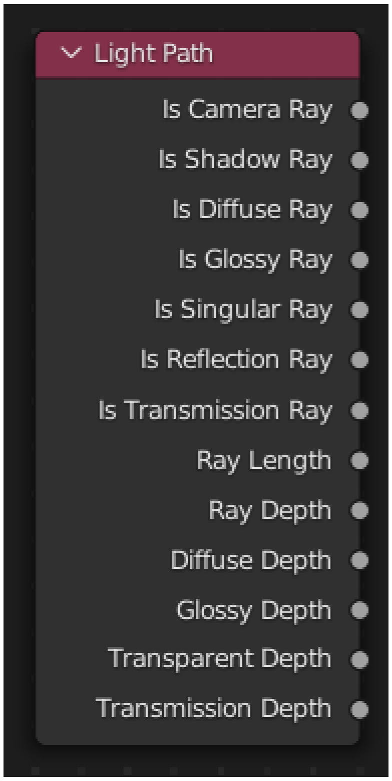
Outputs

Fresnel
: Dielectric Fresnel weight, useful for example for layering diffuse and glossy shaders to create a plastic material. This is like the Fresnel node, except that the input of this node is in the often more convenient 0.0 to 1.0 range.

Facing

Weight that blends from the first to the second shader as the surface goes from facing the viewer to viewing it at a grazing angle.

Light Path Node


Light Path
Is Camera Ray
Is Shadow Ray
Is Diffuse Ray
Is Glossy Ray
Is Singular Ray
Is Reflection Ray
Is Transmission Ray
Ray Length
Ray Depth
Diffuse Depth
Glossy Depth
Transparent Depth
Transmission Depth

The *Light Path* node is used to find out for which kind of incoming ray the shader is being executed; particularly useful for non-physically-based tricks. More information about the meaning of each type is in the Light Paths documentation.

Inputs

This node has no inputs.

Properties

This node has no properties.

Outputs

Is Camera Ray
: 1.0 if shading is executed for a camera ray, otherwise 0.0.

Is Shadow Ray

1.0 if shading is executed for a shadow ray, otherwise 0.0.

Is Diffuse Ray

1.0 if shading is executed for a diffuse ray, otherwise 0.0.

Is Glossy Ray

1.0 if shading is executed for a glossy ray, otherwise 0.0.

Is Singular Ray *Cycles Only*

1.0 if shading is executed for a singular ray, otherwise 0.0.

Is Reflection Ray *Cycles Only*

1.0 if shading is executed for a reflection ray, otherwise 0.0.

Is Transmission Ray *Cycles Only*

1.0 if shading is executed for a transmission ray, otherwise 0.0.

Ray Length *Cycles Only*

Distance traveled by the light ray from the last bounce or camera.

Ray Depth

Number of times the ray has been reflected or transmitted on interaction with a surface.

Note

Passing through a transparent shader does not count as a normal "bounce".

Diffuse Depth *Cycles Only*

Number of times the ray has gone through diffuse reflection or transmission.

Glossy Depth *Cycles Only*

Number of times the ray has gone through glossy reflection or transmission.

Transparent Depth *Cycles Only*

Number of times the ray has gone through a transparent surface.

Transmission Depth *Cycles Only*

Number of times the ray has gone through a transmissive surface. A typical use case is to avoid black spots in the render (caused by rays hitting the bounce limit) by switching from a transmissive to a diffuse shader past a certain point. See Mix Shader.

EEVEE Support

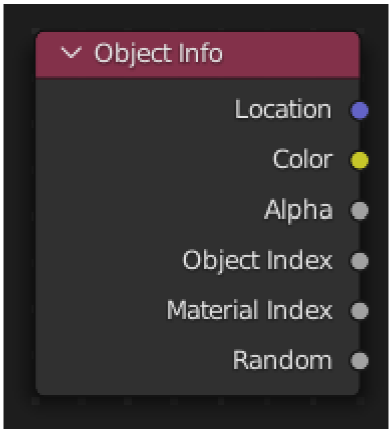
EEVEE has no real concept of rays, but in order to ease the workflow between Cycles and EEVEE, some of the outputs are supported in particular cases.

- *Is Camera*: Supported.
- *Is Shadow*: Supported.
- *Is Diffuse*: Supported.
- *Is Glossy*: Supported.
- *Is Singular*: Not supported. Same as *Is Glossy*.
- *Is Reflection*: Not supported. Same as *Is Glossy*.
- *Is Transmission*: Not supported. Same as *Is Glossy*.
- *Ray Length*: Distance from the camera to the shading point.
- *Ray Depth*: Indicates the current bounce when baking the light cache.
- *Diffuse Depth*: Same as Ray Depth but only when baking diffuse light.
- *Glossy Depth*: Same as Ray Depth but only when baking specular light.
- *Transparent Depth*: Not supported. Defaults to 0.
- *Transmission Depth*: Not supported. Same as *Glossy Depth*.

Note

Is Glossy does not work with Screen Space Reflections/Refractions but does work with reflection planes (whether used with SSR or not).

Object Info Node



The *Object Info* node gives information about the object instance. This can be useful to give some variation to a single material assigned to multiple instances, either manually controlled through the object index, based on the object location, or randomized for each instance. For example a Noise texture can give random colors or a Color Ramp can give a range of colors to be randomly picked from.

Inputs

This node has no inputs.

Properties

This node has no properties.

Outputs

Location

Location of the object in world space.

Color

Object color, same as *Color* in the *Properties* ▸ *Object Properties* ▸ *Viewport Display*.

Alpha

The Alpha Channel component of the object’s viewport display color (see the *Color* output for more details).

Object Index

  Object pass index, same as *Pass Index* in the *Properties* ▸ *Object Properties* ▸ *Relations*.

Material Index

  Material pass index, same as *Pass Index* in the *Properties* ▸ *Material* ▸ *Settings*.

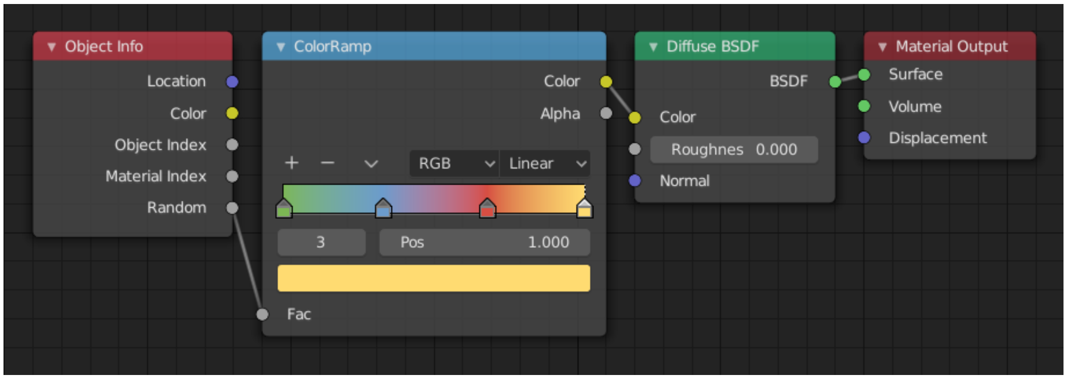
Random

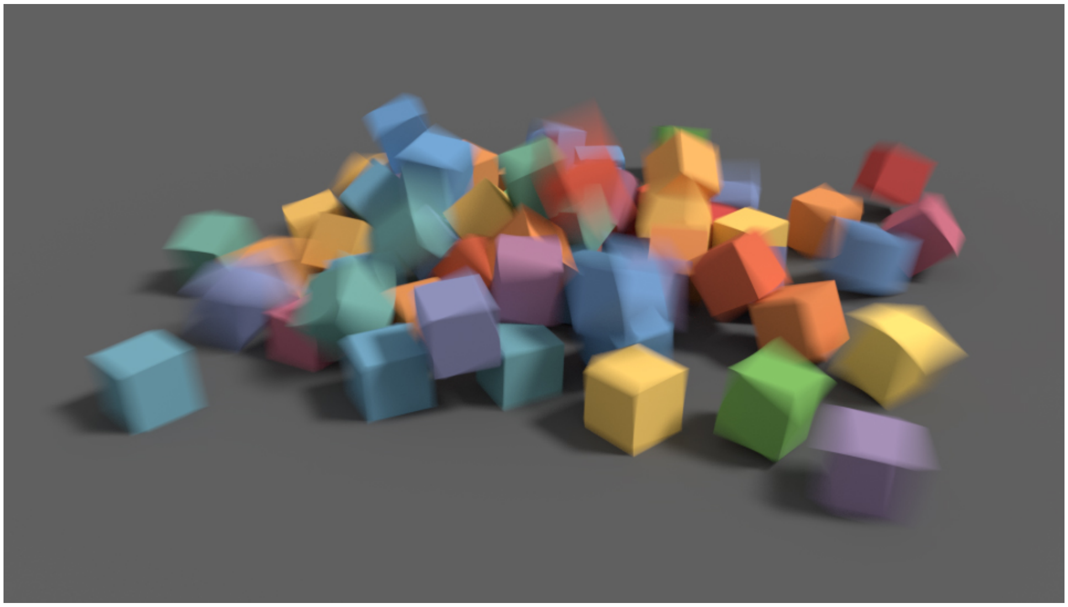
  Random number unique to a single object instance. Output is a Float between 0.0 and 1.0

Note

Note that this node only works for material shading nodes; it does nothing for light and world shading nodes.

Example

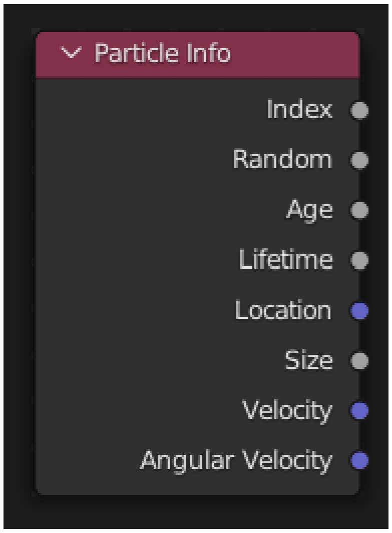

Object Info
Location
Color
Object Index
Material Index
Random
ColorRamp
Color
Alpha
RGB
Linear
3
Pos 1.000
Fac
Diffuse BSDF
BSDF
Color
Roughnes 0.000
Normal
Material Output
Surface
Volume
Displacement



Example blend-file.

Particle Info Node

Cycles Only


Particle Info
Index
Random
Age
Lifetime
Location
Size
Velocity
Angular Velocity

The *Particle Info* node can be used in the material node tree for objects that are used as the instancing objects, when you use *Object* or *Collection* Render mode of a particle system.

This node gives access to the data of the particle that spawned the object instance. It can be useful to give some variation to a single material assigned to multiple instances of instancing object.

Note

This node currently only supports parent particles. Info from child particles is not available.

Inputs

This node has no inputs.

Properties

This node has no properties.

Outputs

Index
    Index number of the particle (from 0 to number of particles).

Random
A random per-particle value in the range from 0 to 1. It can for example be used in combination with a color ramp, to randomize the particle color.

Age
Age of the particle in frames.

Lifetime
Total lifespan of the particle in frames.

Location
Location of the particle.

Size
Size of the particle.

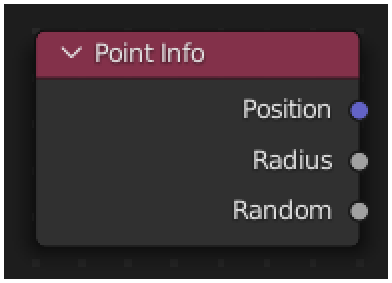
Velocity
Velocity of the particle.

Angular Velocity
Angular velocity of the particle.

Point Info

Cycles Only

The *Point Info* node can be used in the material node tree for point cloud objects and gives access to the data of individual points. It can be useful to give some variation to a single material assigned a point cloud object.



Inputs

This node has no inputs.

Properties

This node has no properties.

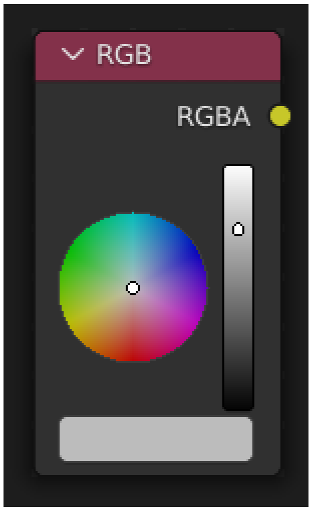
Outputs

Location
    Location of the particle.

Radius
    Size of the particle.

Random
    A random per-point value in the range from 0 to 1. It can for example be used in combination with a color ramp, to randomize the point color.

RGB Node



Inputs

This node has no input sockets.

Properties

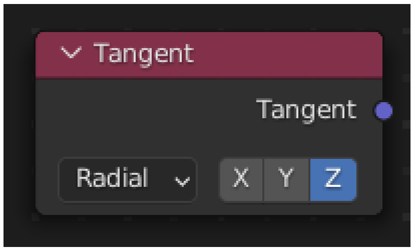
The RGB node uses the color picker widget.

Outputs

Color / RGBA

A single RGBA color value.

Tangent Node



The *Tangent* node generates a tangent direction for the Anisotropic BSDF.

Inputs

This node has no inputs.

Properties

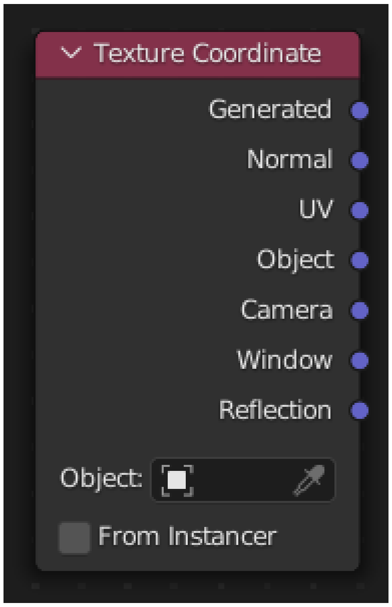
Direction Type
: The tangent direction can be derived from a cylindrical projection around the X, Y, or Z axis (radial), or from a manually created UV Map for full control.

Outputs

Tangent
: The tangent direction vector.

Texture Coordinate Node

The *Texture Coordinate* node is commonly used for the coordinates of textures, typically used as inputs for the *Vector* input for texture nodes.


Texture Coordinate
Generated
Normal
UV
Object
Camera
Window
Reflection
Object:
From Instancer

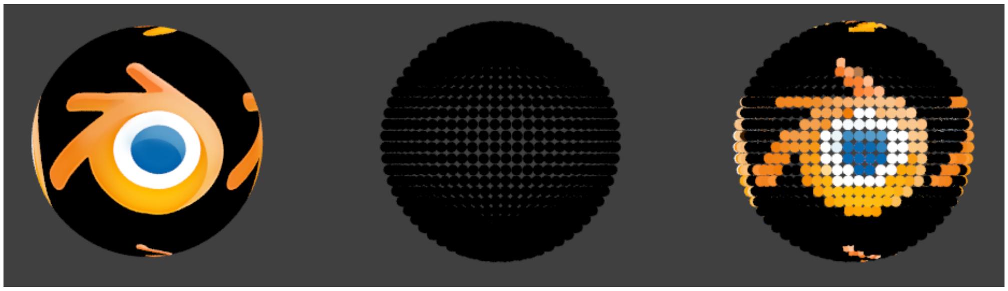
Inputs

This node has no inputs.

Properties

Object
: Specific object to use for object space coordinates. This only affects the *Object* output.

From Instancer *Cycles Only*
: If the object is generated by instancing from vertices or faces, use texture coordinates from instancer. This only affects the *Generated* and *UV* outputs.



From left to right: Sphere with a UV-mapped texture. Small spheres instanced to the faces of the textured sphere using instancing from faces. Small spheres with *From Instancer* enabled, using the UV map of the large sphere.

Note

From Instancer only works with the UV output when the object is instanced, either from particles or from faces.

Outputs

Generated

Automatically-generated texture coordinates from the vertex positions of the mesh without deformation, keeping them sticking to the surface under animation. Range from 0.0 to 1.0 over the bounding box of the undeformed mesh. See Texture Spaces for more information.

Normal

Object space normal, for texturing objects with the texture staying fixed on the object as it transformed. The Normal output can be used on Point and Spot lights. The coordinates will take the rotation of the light into account.

UV

UV texture coordinates from the active render UV map. See UV Mapping for more information.

Note

In order to select UV map other than the active map you must use the UV Map node.

Object

Uses an object as a source for coordinates. Often used with an empty, this is an easy way to place a small image at a given point on the object. This object can also be animated, to move a texture around or through a surface.

Camera

Position coordinate in camera space.

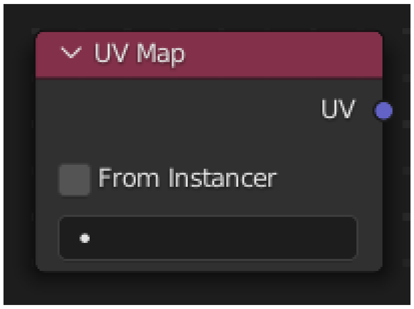
Window

Location of shading point on the screen, ranging from 0.0 to 1.0 from the left to right side and bottom to top of the render. This is well suited for blending two objects.

Reflection

Uses the direction of the reflection vector as coordinates. This is useful for adding reflection maps. You will need this input when using environment maps.

UV Map Node



The *UV Map* node is used to retrieve specific UV maps. Unlike the Texture Coordinate Node which only provides the active UV map, this node can retrieve any UV map belonging to the object using the material.

Inputs

This node has no inputs.

Properties

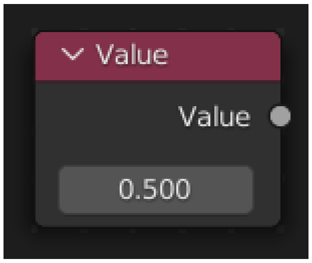
From Instancer *Cycles Only*
  See the From Instancer option of the Texture Coordinate Node.

UV Map
  UV map to use.

Outputs

UV
  UV mapping coordinates from the specified UV map.

Value Node



The *Value Node* is a simple node to input numerical values to other nodes in the tree.

Inputs

This node has no input sockets.

Properties

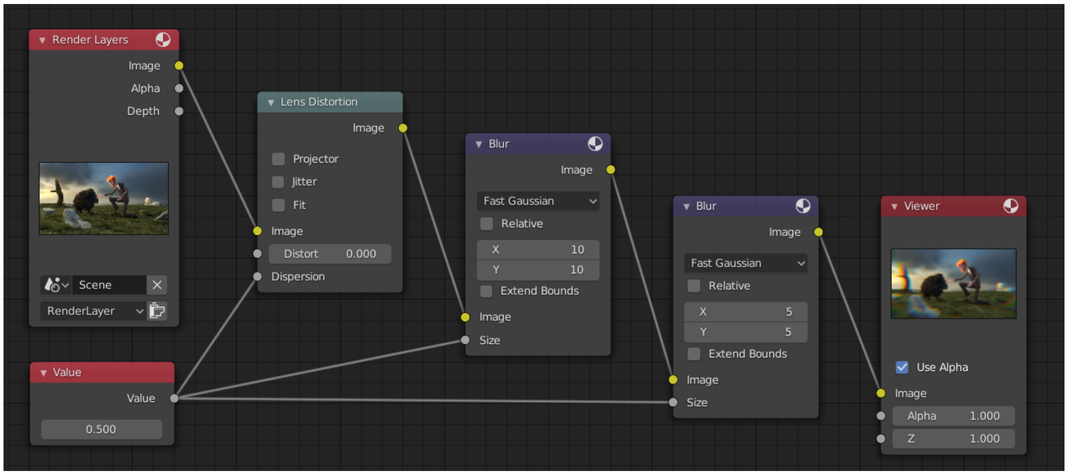
Single numerical value (floating-point).

Outputs

Value
: The value set in the node properties.

Example

In the following example the *Value Node* is used to control multiple values at once, this makes the node a useful organizational tool.


Render Layers
Image
Alpha
Depth
Scene
RenderLayer
Value
Value
0.500
Lens Distortion
Image
Projector
Jitter
Fit
Image
Distort 0.000
Dispersion
Blur
Image
Fast Gaussian
Relative
X 10
Y 10
Extend Bounds
Image
Size
Blur
Image
Fast Gaussian
Relative
X 5
Y 5
Extend Bounds
Image
Size
Viewer
Use Alpha
Image
Alpha 1.000
Z 1.000

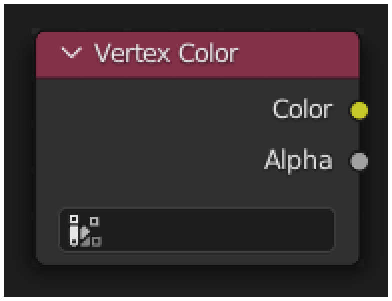
Example of the *Value* node.

Tip

From this you can also make different values proportional to each other by adding a Math Node in between the different links.

Color Attribute Node

The *Color Attribute* node provides access to Color Attributes as well as their alpha value.



Inputs

This node has no inputs.

Properties

Color Attribute
: The target Color Attribute. The listed Color Attributes are those of the mesh of the active object. If the active object has no mesh, a warning will be displayed. If the property is marked in red, it means the Color Attribute is not available in the mesh of the active object, but it may be available in other meshes of objects that share this material!

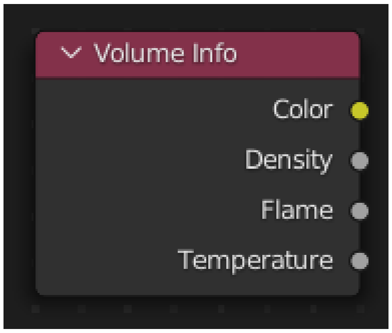
Outputs

Color
: Standard color output.

Alpha
: Alpha value.

Volume Info Node

The *Volume Info* node provides information about *Smoke Domains*.



Inputs

This node has no inputs.

Properties

This node has no properties.

Outputs

Color

Gives the color of the smoke inside the Fluid Domain. The color and vector outputs are the same. The Factor output is an average of the channels.

Density

Gives a scalar defining the density of any smoke inside the Fluid Domain.

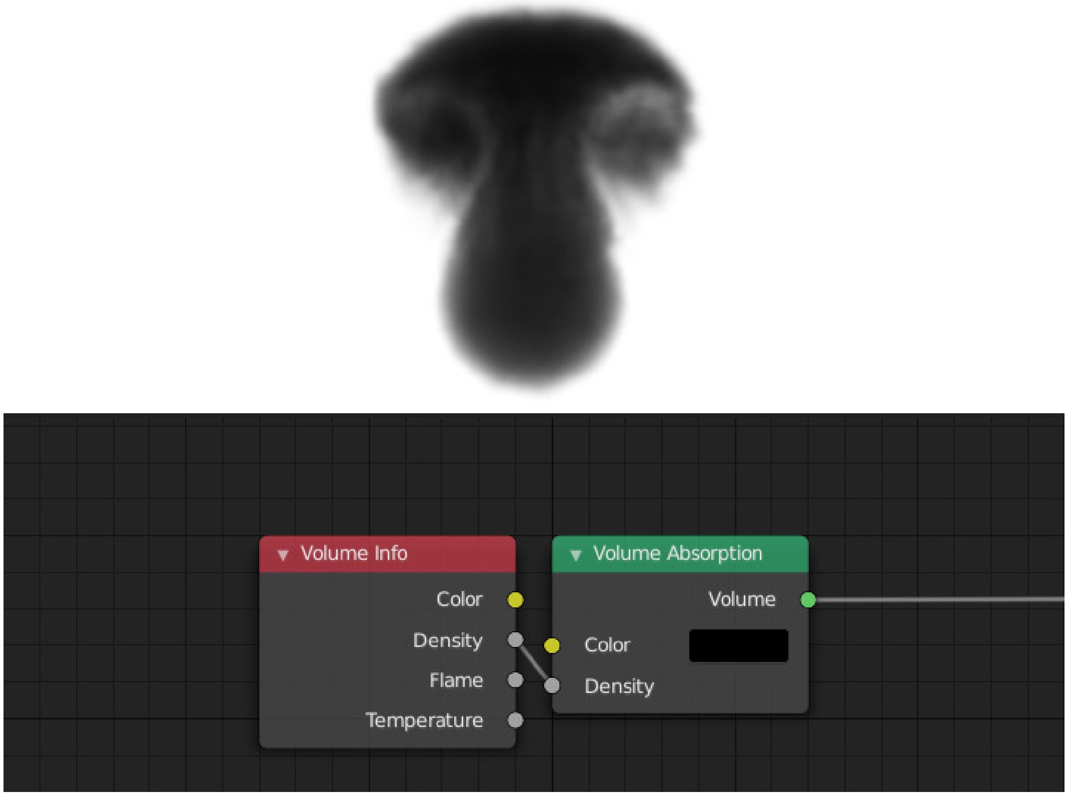
Flame

Gives a scalar defining the density of any fire inside the Fluid Domain. All three outputs are the same.

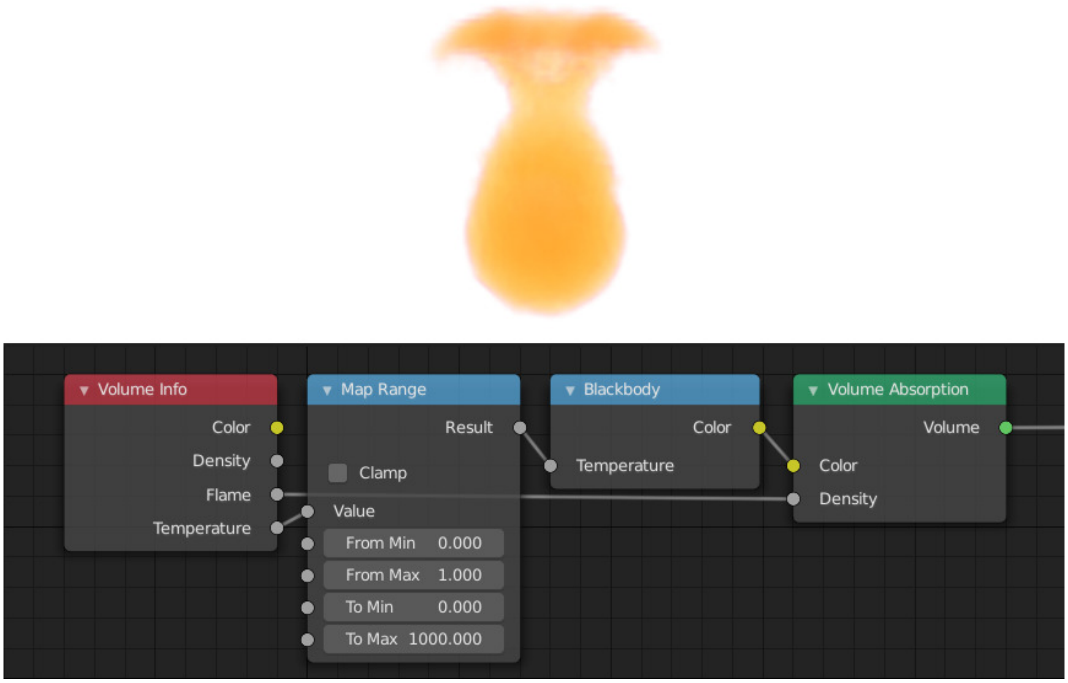
Temperature

Gives a scalar defining the temperature of the volume. Values in the range 0 - 1 map to 0 - 1000 kelvin. This may be used to render physically-based fire with the Blackbody or Principled Volume shaders. All three outputs are the same.

Example


Volume Info
Color
Density
Flame
Temperature
Volume Absorption
Volume
Color
Density

Smoke density.

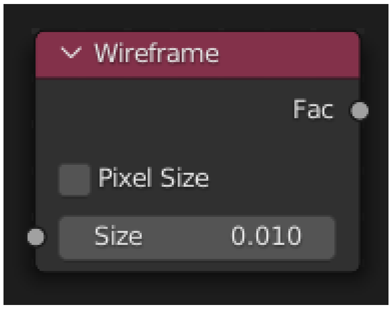

Volume Info
Color
Density
Flame
Temperature
Map Range
Result
Clamp
Value
Blackbody
Color
Temperature
Volume Absorption
Volume
Color
Density

From Min 0.000
From Max 1.000
To Min 0.000
To Max 1000.000

Computing the color of fire using the *Blackbody* node. Since the *Blackbody* node expects its input in Kelvin, the temperature output has to be remapped first.

Wireframe Node

The Wireframe node is used to retrieve the edges of an object as it appears to Cycles. As meshes are triangulated before being processed by Cycles, topology will always appear triangulated when viewed with the Wireframe node.



☐ Pixel Size

Size 0.010

Inputs

This node has no inputs.

Properties

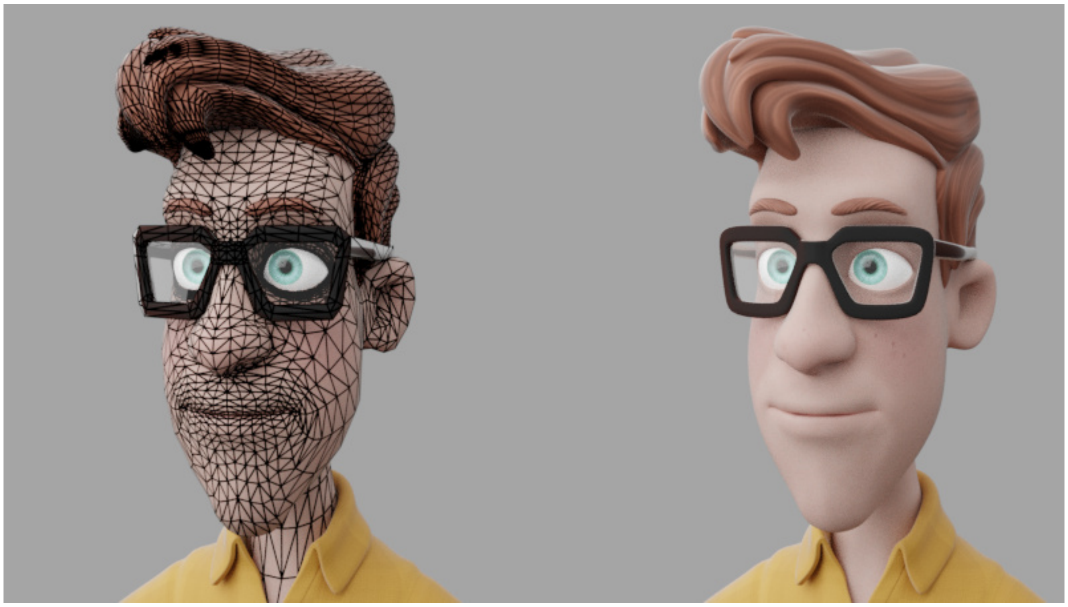
Pixel Size
: When enabled, the size of edge lines is set in screen space.

Size
: Thickness of the edge lines.

Outputs

Factor
: Black-and-white mask showing white lines representing edges according to the object’s Topology.

Examples



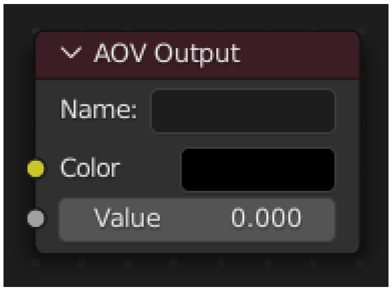
Using the Wireframe node to showcase the topology of a mesh.

Output

Output nodes are the final node in every node tree. Although you can add more than one, only one will be used (indicated by a colored or darkened header). Output nodes are always preceded by Shaders except in the case of the Displacement of a Material Output.

- AOV Output Node
- Material Output Node
- Light Output Node
- World Output Node

AOV Output Node



Shader AOVs (Arbitrary Output Variables) provide custom render passes for arbitrary shader node components. As an artist this can be a good way to debug or tweak very fine details of a scene in post-processing. To use shader AOVs create the pass in the Shader AOV panel then reference that pass with the *AOV Output* shading node. Shader AOVs can be added or removed in the *Shader AOV* panel.

Tip

The *AOV Output* node can be used in Material and World shader nodes.

Inputs

Color

Output a color variable; as the name suggest can be used for a color but also a normal value.

Value

Output a single numerical value.

Properties

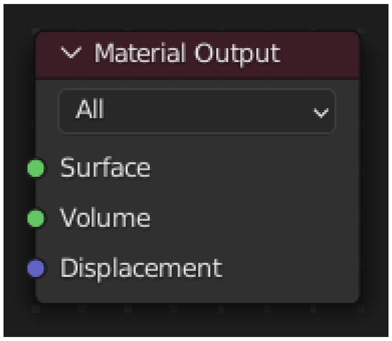
Name

The name of the render pass to assign the input value to. This property has the same *Name* that is specified in the Shader AOV panel.

Outputs

This node has no outputs.

Material Output Node



The *Material Output* node is used to output surface material information to a surface object.

Inputs

Surface
: Shading for the surface of the object.

Volume
: Shading for the volume inside the object.

Displacement
: Used to create bump mapping or actual subdivided displacement.

Thickness *EEVEE*
: Used to approximate the inner geometry structure of the object without heavy computation. This is currently used for Subsurface Scattering, Translucent BSDF, Refraction BSDF, and the nodes containing these effects.

  If no value is plugged into the output node, a default thickness based on the smallest dimension of the object is used. If a value is connected it will be used as object space thickness (i.e. scaled by object transform). A value of zero will disable the thickness approximation and treat the object as having only one interface.

  This output is only used by the EEVEE render engine.

Note

- The thickness is used to skip the inner part of the object.
- Refraction will not refract objects inside the thickness distance.
- Shadow casting object will not cast shadow within the thickness distance.

Tip

- For large or compound meshes (e.g. vegetation), the thickness should be set to the thickness of individual parts (e.g. leaves, grass blades).
- Thickness can be baked to textures or custom attributes for more accurate result.

See also

Thickness Mode – controls how the thickness value is used.

Properties

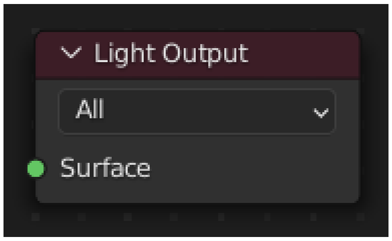
Target
: Render engine the input shaders are used for. By default shaders are shared between Cycles and EEVEE, with multiple output nodes specialized shader setups can be created for each.

Outputs

This node has no outputs.

Light Output Node

The *Light Output* node is used to output light information to a light object.



Inputs

Surface

Shading for the surface of the light object.

Properties

Target

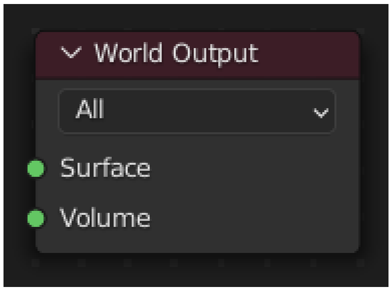
Render engine the input shaders are used for. By default shaders are shared between Cycles and EEVEE, with multiple output nodes specialized shader setups can be created for each.

Outputs

This node has no outputs.

World Output Node

The *World Output* node is used to output light a color information to the scene's World.



Inputs

Surface

The appearance of the environment, usually preceded by a Background shader.

Volume

Used to add volumetric effects to the world. See the shaders Volume Absorption and Volume Scatter for more information.

> **Note**
>
> It is not possible to have an HDR and volumetric due to the fact that HDR's are assumed to be an infinite distance from the camera.

Properties

Target

Render engine the input shaders are used for. By default shaders are shared between Cycles and EEVEE, with multiple output nodes specialized shader setups can be created for each.

Outputs

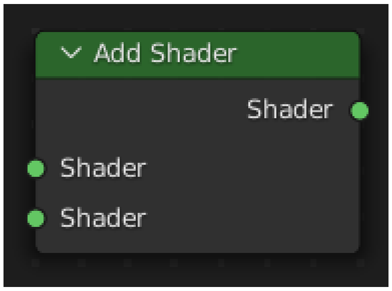
This node has no outputs.

Shader

- Add Shader
- Background
- Diffuse BSDF
- Emission
- Glass BSDF
- Glossy BSDF
- Hair BSDF
- Holdout
- Mix Shader
- Metallic BSDF
- Principled BSDF
- Principled Hair BSDF
- Principled Volume
- Ray Portal BSDF
- Refraction BSDF
- Specular BSDF
- Subsurface Scattering
- Toon BSDF
- Translucent BSDF
- Transparent BSDF
- Sheen BSDF
- Volume Absorption
- Volume Scatter

Add Shader

The *Add* node is used to add two *Shaders* together.


Add Shader
Shader
Shader
Shader

Inputs

Shaders
    Standard shader inputs.

Properties

This node has no properties.

Outputs

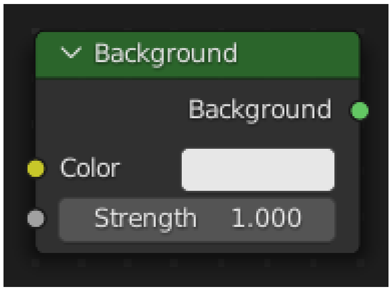
Shader
    Standard shader output.

Example

A mix of a glossy and a diffuse shader makes a nice ceramic material.

Background

The *Background* shader node is used to add background light emission. This node should only be used for the world surface output.


Background
Background
Color
Strength 1.000

Inputs

Color
    Color of the emitted light.

Strength
    Strength of the emitted light.

Properties

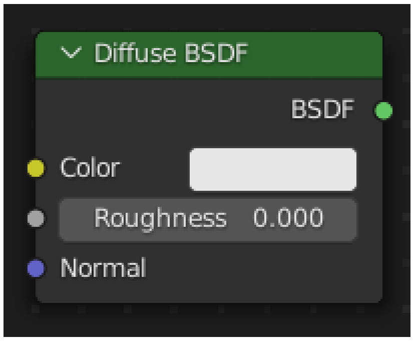
This node has no properties.

Outputs

Background
    Standard shader output.

Diffuse BSDF

The *Diffuse* BSDF node is used to add Lambertian and Oren-Nayar diffuse reflection.



Inputs

Color
: Color of the surface, or physically speaking, the probability that light is reflected or transmitted for each wavelength.

Roughness *Cycles Only*
: Surface roughness; 0.0 gives standard Lambertian reflection, higher values activate the Oren-Nayar BSDF.

Normal
: Normal used for shading; if nothing is connected the default shading normal is used.

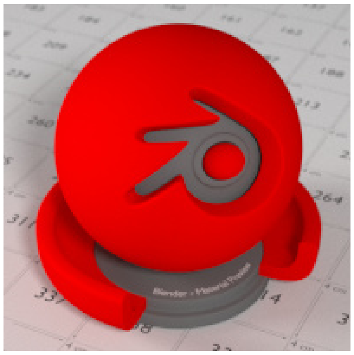
Properties

This node has no properties.

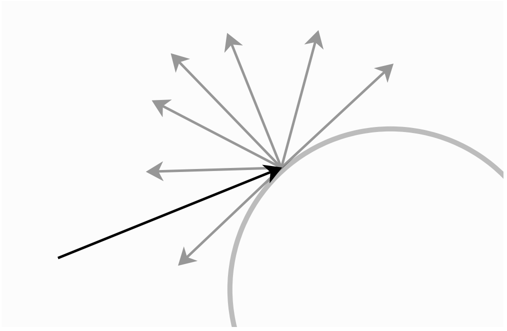
Outputs

BSDF
: Standard shader output.

Examples



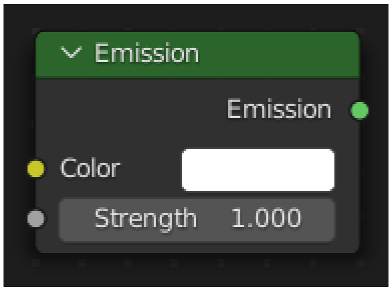
Lambertian reflection.



Diffuse shader behavior.

Oren-Nayar reflection.

Emission



The *Emission* node is used to add Lambertian emission shader. This can for example, be used for material and light surface outputs.

Light strength for point, spot and area lights is specified in Watts.

Sun lights are specified in Watts/m^2, which require much smaller values like 1 W/m^2. This can be confusing, but specifying strength in Watts would not have been convenient; the real sun for example has strength 384.6×10^{24}W. Emission shaders on meshes are also in Watts/m^2.

Inputs

Color
    Color of the emitted light.

Strength
    Strength of the emitted light. For point and area lights, the unit is Watts. For materials, a value of 1.0 will ensure that the object in the image has the exact same color as the Color input, i.e. make it ‘shadeless’.

Properties

This node has no properties.

Outputs

Emission

The Emission shader output can both be plugged into the *Surface Input* as well as the *Volume Input* of the Material Output node.

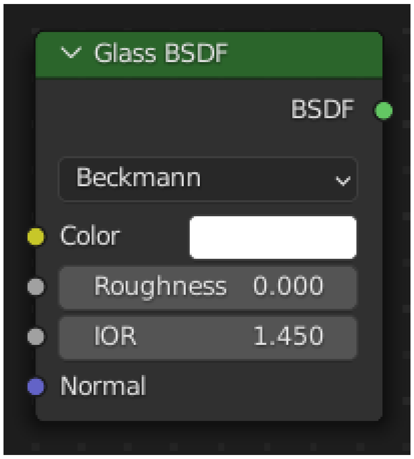
Examples

Emission shader, with strength at 1.0.

Emission shader, with strength at 3.0.

Glass BSDF

The *Glass* BSDF is used to add a Glass-like shader mixing refraction and reflection at grazing angles. Like the transparent shader, only pure white will make it transparent. The glass shader tends to cause noise due to caustics. Since the Cycles path tracing integrator is not very good at rendering caustics, it helps to combine this with a transparent shader for shadows; for more details see here.



Beckmann

Roughness 0.000

IOR 1.450

Inputs

Color

Color of the surface, or physically speaking, the probability that light is transmitted for each wavelength.

Roughness

Influences sharpness of the refraction; perfectly sharp at 0.0 and smoother with higher values.

IOR

Index of refraction (IOR) defining how much the ray changes direction. At 1.0 rays pass straight through like transparent; higher values give more refraction.

Normal

Normal used for shading.

Properties

Distribution

Microfacet distribution to use.

GGX::	GGX microfacet distribution.
Multiscatter GGX::	Takes multiple scattering events between microfacets into account. This gives more energy conserving results, which would otherwise be visible as excessive darkening.
Beckmann::	*Cycles Only* Beckmann microfacet distribution.

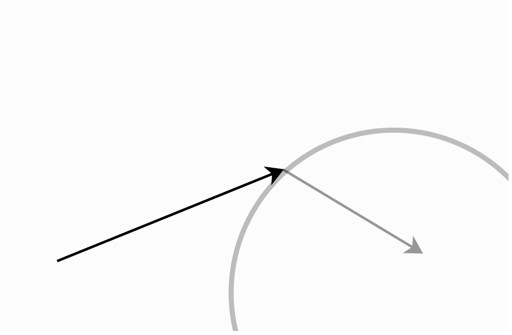
Outputs

BSDF

Standard shader output.

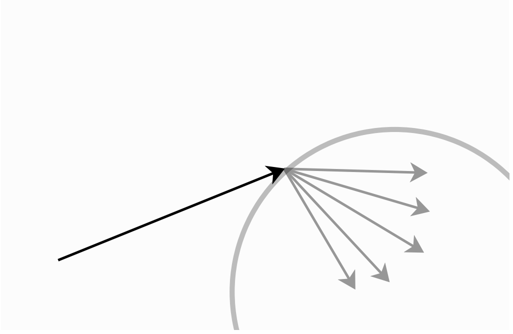
Examples

Sharp Glass example.



Sharp Glass behavior.

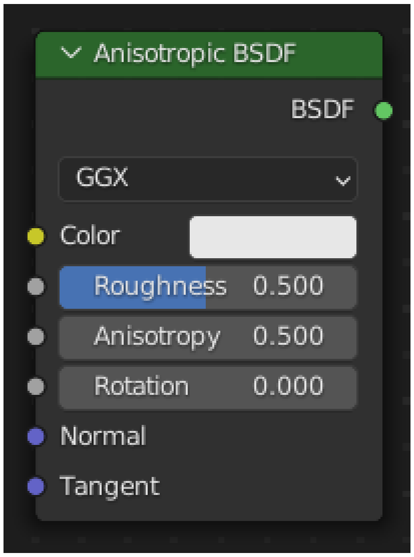
Rough Glass example.



Rough Glass behavior.

Glossy BSDF

The *Glossy* BSDF node is used to add reflection with microfacet distribution, used for materials such as metal or mirrors.



Roughness 0.500
Anisotropy 0.500
Rotation 0.000

Inputs

Color

Color of the surface, or physically speaking, the probability that light is reflected for each wavelength.

Roughness

Sharpness of the reflection; perfectly sharp at 0.0 and smoother with higher values.

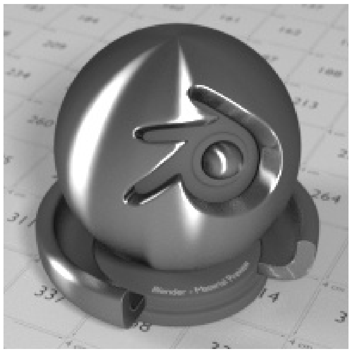
Anisotropy *Cycles Only*

Controls the amount the reflection stretches the reflection along the surface of the material. A value of 0.0 results in no anisotropic reflections. Higher values give elongated highlights orthogonal to the tangent direction; negative values give highlights shaped along the tangent direction.

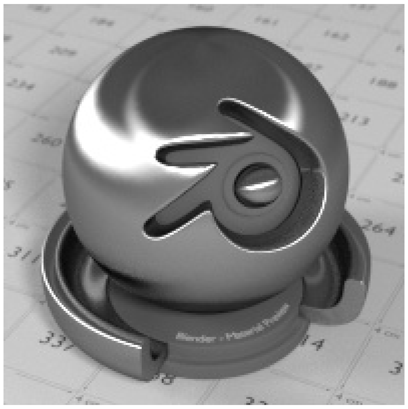
This is a phenomenon know as “Anisotropic Reflections” which is often seen in metallic materials.

Rotation

Rotation of the anisotropic tangent direction. Value 0.0 equals 0° rotation, 0.25 equals 90° and 1.0 equals 360° = 0°. This can be used to texture the tangent direction.



Anisotropic rotation on 0.



Anisotropic rotation on 0.25 (90°).

Normal

Normal used for shading; if nothing is connected the default shading normal is used.

Tangent

Tangent used for shading; if nothing is connected the default shading tangent is used.

Properties

Distribution

Microfacet distribution to use.

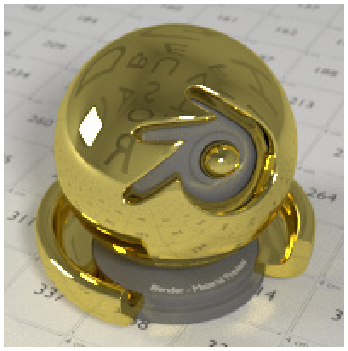
GGX::	GGX microfacet distribution.
Multiscatter GGX::	Takes multiple scattering events between microfacets into account. This gives more energy conserving results, which would otherwise be visible as excessive darkening.

Beckmann:: *Cycles Only* Beckmann microfacet distribution.

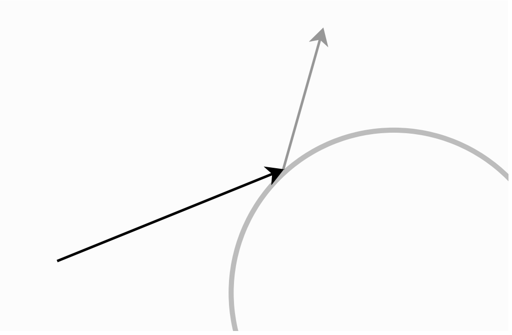
Outputs

BSDF
Standard shader output.

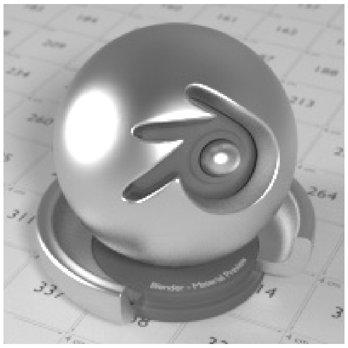
Examples



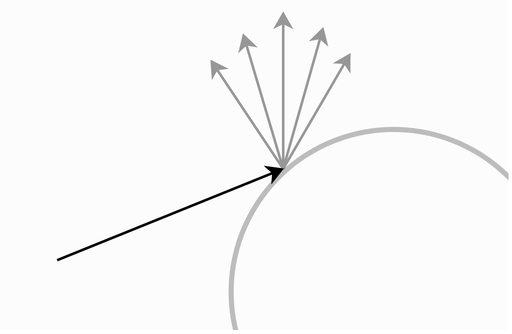
Sharp Glossy example.



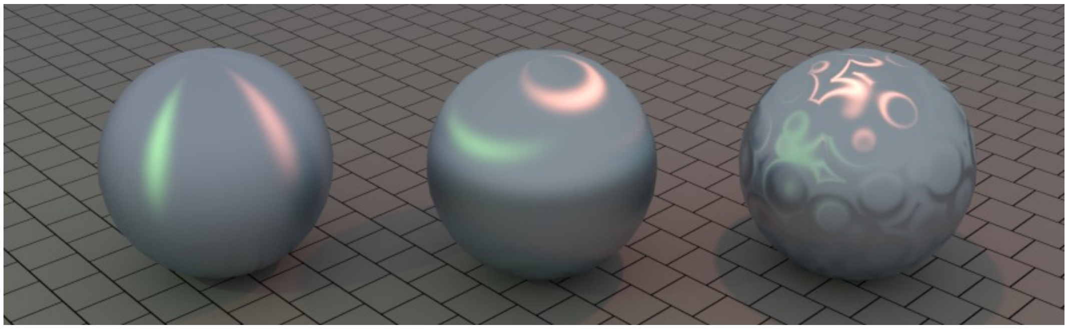
Sharp Glossy behavior.



Rough Glossy example.



Rough Glossy behavior.

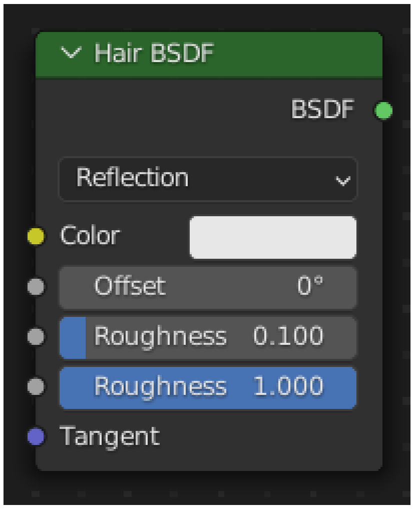


Anisotropic shading with 0° rotation, 90° rotation and textured rotation of the tangent direction. Example blend-file.

Hair BSDF

Cycles Only

The *Hair* BSDF node is used to add shading for Hair.


Hair BSDF
BSDF
Color
Tangent

Reflection

Offset 0°

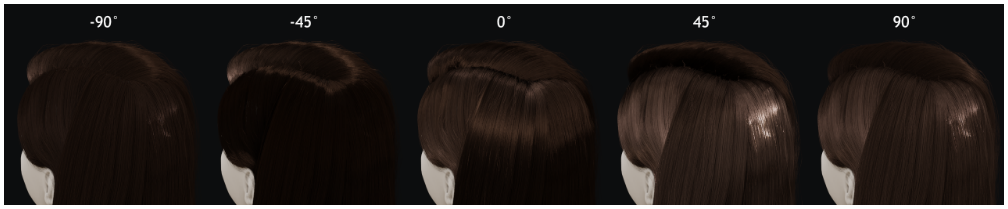
Roughness 0.100

Roughness 1.000

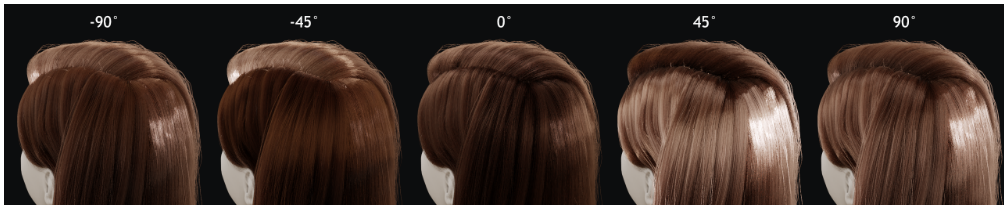
Inputs

Color
: Color of the hair.

Offset
: Controls the way the light is rotated (angular shift) for the reflection/transmission.



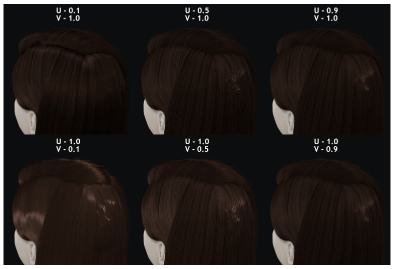
Reflection Offset.



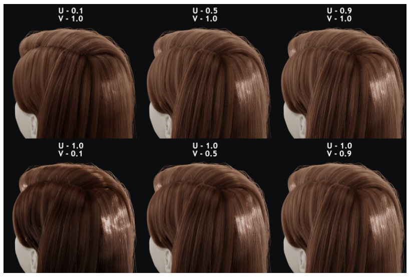
Transmission Offset.

Roughness U/V

Controls the roughness in the direction light is skewed, and perpendicular to it.


U - 0.1
V - 1.0
U - 0.5
V - 1.0
U - 0.9
V - 1.0
U - 1.0
V - 0.1
U - 1.0
V - 0.5
U - 1.0
V - 0.9

Roughness when using the Reflection Component.


U - 0.1
V - 1.0
U - 0.5
V - 1.0
U - 0.9
V - 1.0
U - 1.0
V - 0.1
U - 1.0
V - 0.5
U - 1.0
V - 0.9

Roughness when using the Transmission Component.

Tangent

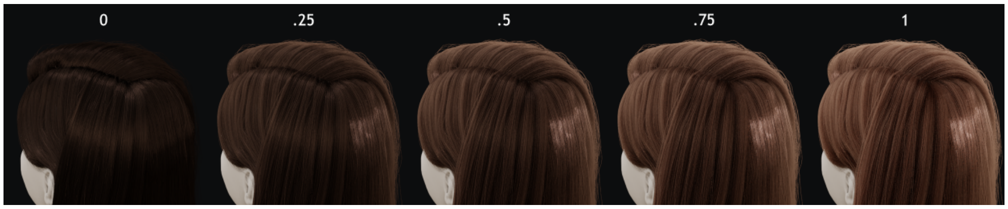
Input tangent.

Properties

Component

There are two components that can be used to control the look of the hair. Usually you are going to want each of these and use a Mix Node.

Reflection::	The light that bounces off the surface of the hair.
Transmission::	The light that passes through the hair and exits on the other side.

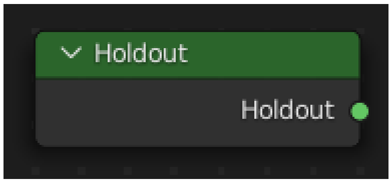


With Mix node: 0 is full Reflection, 1 is full Transmission.

Outputs

BSDF
: Standard shader output.

Holdout



The *Holdout* shader node is used to create a “hole” in the image with zero alpha transparency, which is useful for compositing (see Alpha Channel).

Inputs

This node has no inputs.

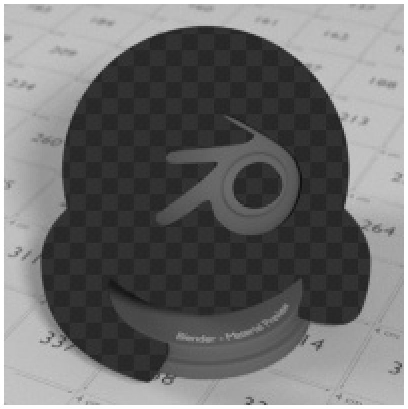
Properties

This node has no properties.

Outputs

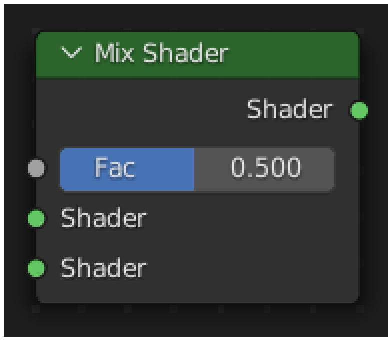
Holdout
    Standard shader output.

Examples



The checkered area is a region with zero alpha.

Mix Shader



The *Mix* node is used to mix two shaders together. Mixing can be used for material layering, where the *Factor* input may, for example, be connected to a *Blend Weight* node.

Inputs

Shader
: Shaders to mix, such that incoming rays hit either with the specified probability in the *Factor* socket.

Factor
: Blend weight to use for mixing two shaders; at zero it uses the first shader entirely and at one the second shader.

Properties

This node has no properties.

Outputs

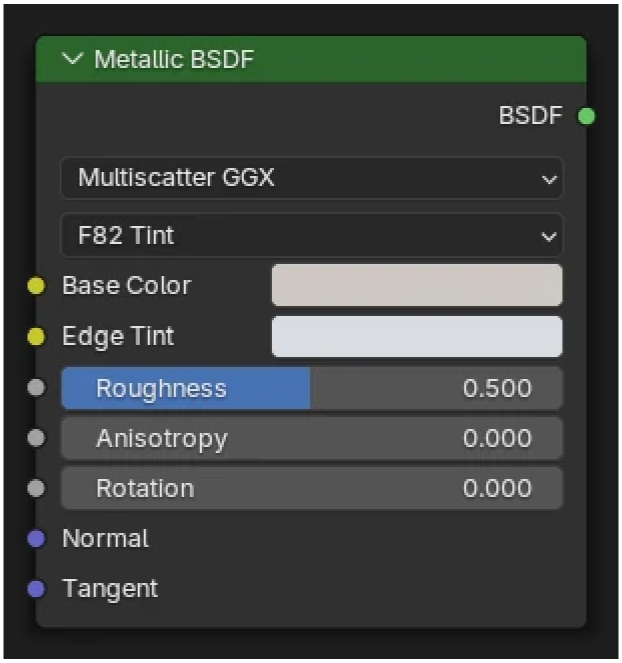
Shader
: Standard shader output.

Examples

A mix of a glossy and a diffuse shader makes a nice ceramic material.

Metallic BSDF

The *Metallic* BSDF node is used to recreate the appearance of metals.


Metallic BSDF
BSDF
Multiscatter GGX
F82 Tint
Base Color
Edge Tint
Roughness 0.500
Anisotropy 0.000
Rotation 0.000
Normal
Tangent

Inputs

F82 Tint

Base Color
: Color of the material when viewed straight on.

Edge Tint
: Color of the material when viewed at a 82° angle.

Physical Conductor

IOR
: Refractive index per color channel. This is the real part of a complex refractive index, scientifically denoted as n.

Extinction

Extinction coefficients per color channel. This is the imaginary part of a complex refractive index, scientifically denoted as k.

Common

Roughness

Sharpness of the reflection; perfectly sharp at 0.0 and smoother with higher values.

Anisotropy *Cycles Only*

Amount of anisotropy. Higher values give elongated highlights along the tangent direction; negative values give highlights shaped perpendicular to the tangent direction.

Rotation *Cycles Only*

Rotates the direction of anisotropy, with 1.0 going full circle.

Compared to the *Glossy BSDF* node, the direction of highlight elongation is rotated by 90°. Add 0.25 to the value to correct.

Normal

Normal used for shading; if nothing is connected the default shading normal is used.

Tangent

Tangent used for shading; if nothing is connected the default shading tangent is used.

Properties

Distribution

Microfacet distribution to use.

GGX::	GGX microfacet distribution.
Multiscatter GGX::	

	Takes multiple scattering events between microfacets into account. This gives more energy conserving results, which would otherwise be visible as excessive darkening.
Beckmann::	*Cycles Only* Beckmann microfacet distribution.

Fresnel Type

Models for describing the metal's appearance, by specifying the apparent color or the physical IOR.

F82 Tint::	Uses the Adobe F82-Tint formula for the metallic fresnel. This allows for artist friendly control of the color near the edge of the material to simulate a complex IOR.
Physical Conductor::	Accepts Complex IOR measurements from real world metals to replicate a more accurate rendering of metals than the *F82 Tint* Fresnel type. Complex IOR values can be found from sources like the Physically Based database for CG artists and Refractive Index nk database.

Outputs

BSDF

Standard shader output.

Examples

F82 Tint

Material	Titanium (Default)	Aluminum	Copper	Gold
Base Color	0.617, 0.576, 0.540	0.911, 0.912, 0.917	0.972, 0.694, 0.486	1.000, 0.735, 0.353
Edge Tint	0.695, 0.726, 0.770	0.848, 0.877, 0.916	0.961, 0.969, 0.942	0.993, 1.000, 1.000

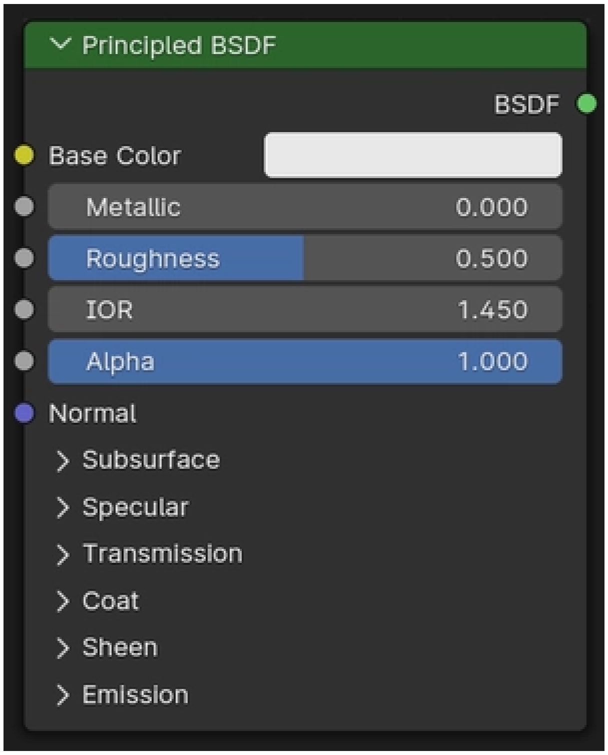
Physical Conductor

Material	Titanium (Default)	Aluminum	Copper	Gold
IOR	2.757, 2.512, 2.231	1.333, 0.945, 0.582	0.235, 0.729, 1.369	0.000, 0.470, 1.439
Extinction	3.867, 3.404, 3.009	7.434, 6.340, 5.181	5.666, 2.562, 2.227	182.6, 2.189, 1.660

Principled BSDF

The *Principled* BSDF that combines multiple layers into a single easy to use node. It can model a wide variety of materials.

It is based on the OpenPBR Surface shading model, and provides parameters compatible with similar PBR shaders found in other software, such as the Disney and Standard Surface models. Image textures painted or baked from software like Substance Painter may be directly linked to the corresponding input in this shader.



Principled BSDF

BSDF

Base Color

Metallic 0.000

Roughness 0.500

IOR 1.450

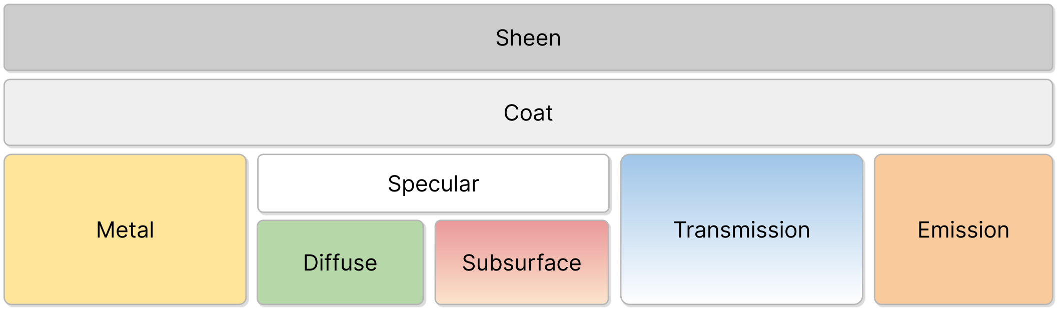
Alpha 1.000

Normal

Subsurface
Specular
Transmission
Coat
Sheen
Emission

Layers

The base layer is a mix between metal, diffuse, subsurface, and transmission components. Most materials will use one of these components, though it is possible to smoothly mix between them.


Sheen
Coat
Metal
Specular
Diffuse
Subsurface
Transmission
Emission

The metal component is opaque and only reflect lights. Diffuse is fully opaque, while subsurface also involves light scattering just below the surface. Both diffuse and subsurface sit below a specular layer. The transmission component includes both specular reflection and refraction.

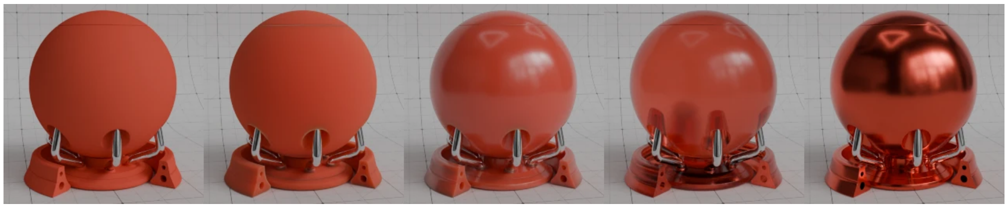
On top of all base layers there is an optional glossy coat. And finally the sheen layer sits on top of all other layers, to add fuzz or dust.

Light emission can also be added. Light emits from below the coat and sheen layers, to model for example emissive displays with a coat or dust.

Inputs

Base Color

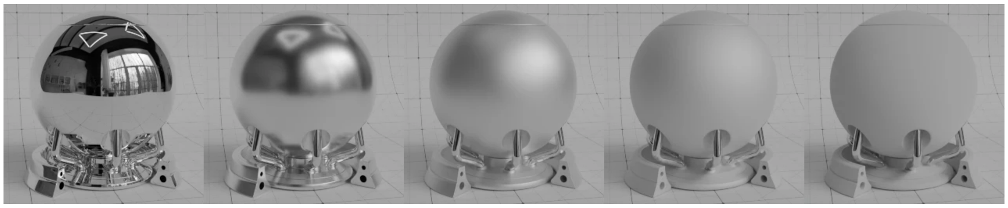
Overall color of the material used for diffuse, subsurface, metal and transmission.



Same base color for multiple materials types

Roughness

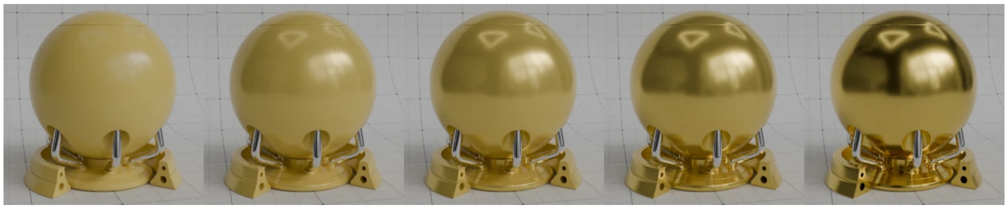
Specifies microfacet roughness of the surface for specular reflection and transmission. A value of 0.0 gives a perfectly sharp reflection, while 1.0 gives a diffuse reflection.



Roughness from 0.0 to 1.0

Metallic

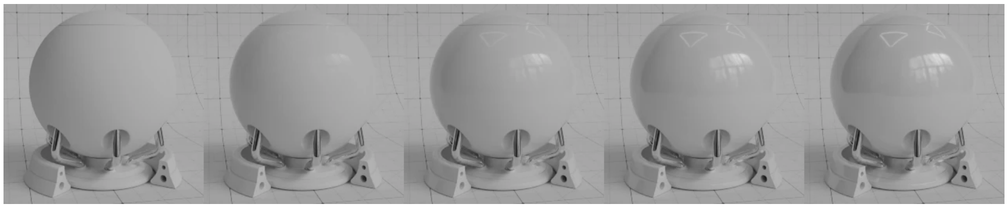
Blends between a dielectric and metallic material model. At 0.0 the material consists of a diffuse or transmissive base layer, with a specular reflection layer on top. A value of 1.0 gives a fully specular reflection tinted with the base color, without diffuse reflection or transmission.



Metallic from 0.0 to 1.0

IOR

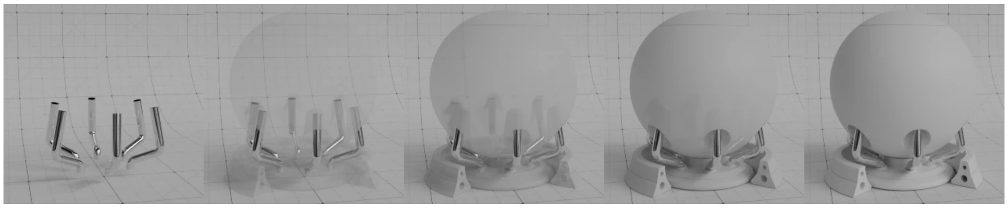
Index of refraction (IOR) for specular reflection and transmission. For most materials, the IOR is between 1.0 (vacuum and air) and 4.0 (germanium). The default value of 1.5 is a good approximation for glass.



IOR from 1.0 to 2.0

Alpha

Controls the transparency of the surface, with 1.0 fully opaque. Usually linked to the Alpha output of an Image Texture node.



Alpha from 0.0 to 1.0

Normal

Controls the normals of the base layers.

Diffuse

Roughness *Cycles Only*

Surface roughness; 0.0 gives standard Lambertian reflection, higher values activate the Oren-Nayar BSDF.

Subsurface

Subsurface scattering is used to render materials such as skin, milk and wax. Light scatters below the surface to create a soft appearance.

Method

Rendering method to simulate Subsurface scattering.

Christensen-Burley::

An approximation to physically-based volume scattering. This method is less accurate than *Random Walk* however, in some situations this method will resolve noise faster.

Random Walk::

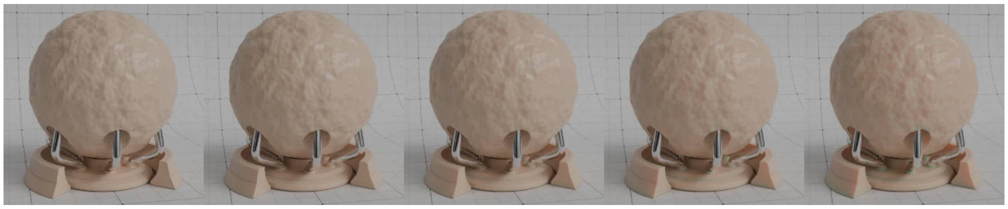
Cycles Only Provides accurate results for thin and curved objects. Random Walk uses true volumetric scattering inside the mesh, which means that it works best for closed meshes. Overlapping faces and holes in the mesh can cause problems.

Random Walk (Skin)::

Cycles Only Random walk method optimized for skin rendering. The radius is automatically adjusted based on the color texture, and the subsurface entry direction uses a mix of diffuse and specular transmission with custom IOR. This tends to retain greater surface detail and color and matches measured skin more closely.

Weight

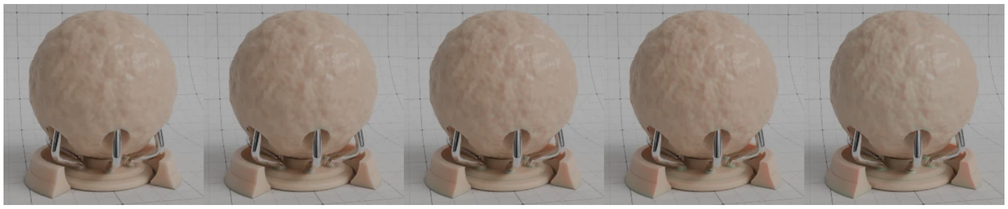
Blend between diffuse surface and subsurface scattering. Typically should be zero or one (either fully diffuse or subsurface).



Weight from 0.0 to 1.0

Radius

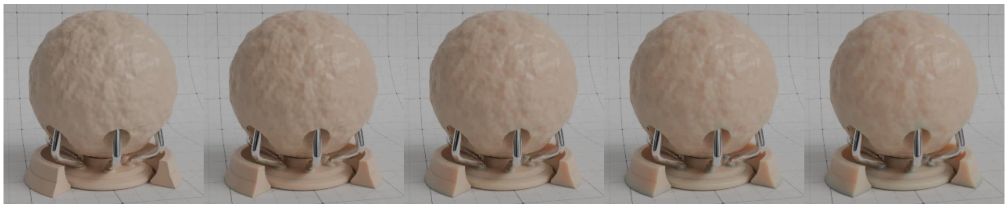
Average distance that light scatters below the surface. Higher radius gives a softer appearance, as light bleeds into shadows and through the object. The scattering distance is specified separately for the RGB channels, to render materials such as skin where red light scatters deeper. The X, Y and Z values are mapped to the R, G and B values, respectively.



Radius from white to red

Scale

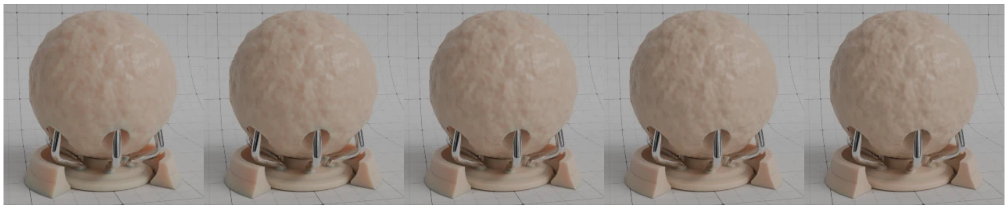
Scale applied to the radius.



Scale from 0 cm to 50 cm

IOR *Cycles Only*

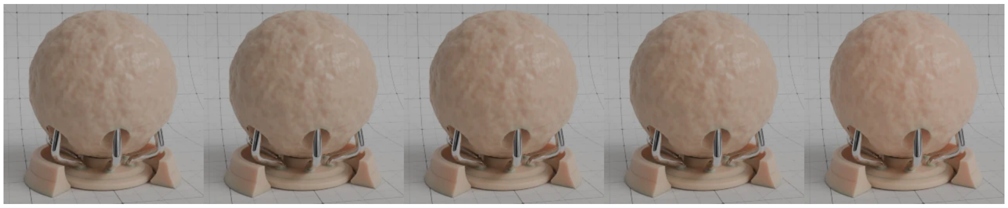
Index of refraction (IOR) used for rays that enter the subsurface component. This may be set to a different value than the global IOR to simulate different layers of skin.



IOR from 1.0 to 2.0

Anisotropy *Cycles Only*

Directionality of volume scattering within the subsurface medium. Zero scatters uniformly in all directions, with higher values scattering more strongly forward. For example, skin has been measured to have an anisotropy of 0.8.



Anisotropy from 0.0 to 1.0

Specular

Controls for both the metallic component and specular layer on top of diffuse and subsurface.

Distribution

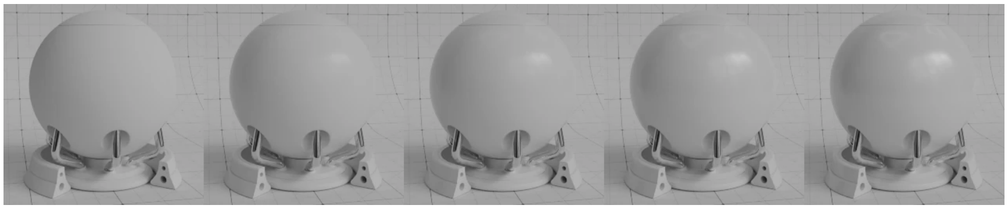
Microfacet distribution to use.

GGX::	A method that is faster than *Multiple-scattering GGX* but is less physically accurate.
Multiscatter GGX::	Takes multiple scattering events between microfacets into account. This gives more energy conserving results, which would otherwise be visible as excessive darkening.

IOR Level

Adjustment to the IOR to increase or decrease intensity of the specular layer. 0.5 means no adjustment, 0 removes all reflections, 1 doubles them at normal incidence.

This input is designed for conveniently texturing the IOR and amount of specular reflection.



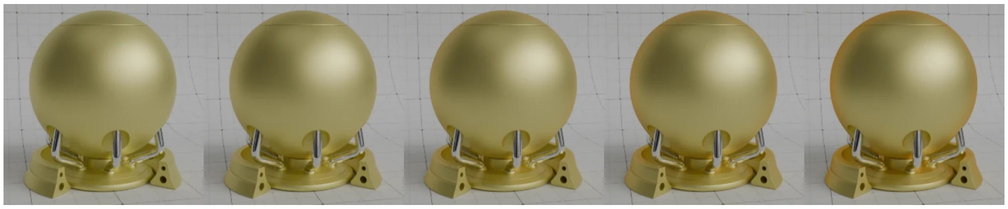
IOR level from 0.0 to 1.0

Tint

Color tint for specular and metallic reflection.

For non-metallic tints provides artistic control over the color specular reflections at normal incidence, while grazing reflections remain white. In reality non-metallic specular reflection is fully white.

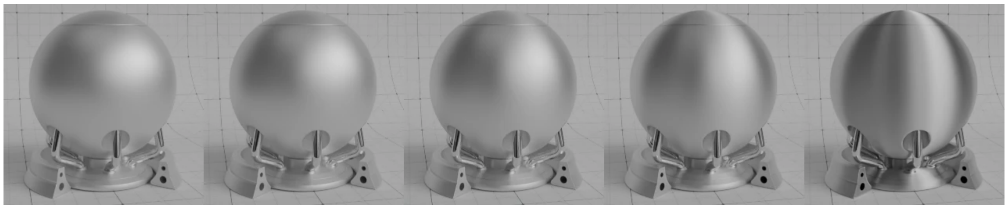
For metallic materials tints the edges to simulate complex IOR as found in materials such as gold or copper.



Tint from white to orange

Anisotropic *Cycles Only*

Amount of anisotropy for specular reflection. Higher values give elongated highlights along the tangent direction; negative values give highlights shaped perpendicular to the tangent direction.

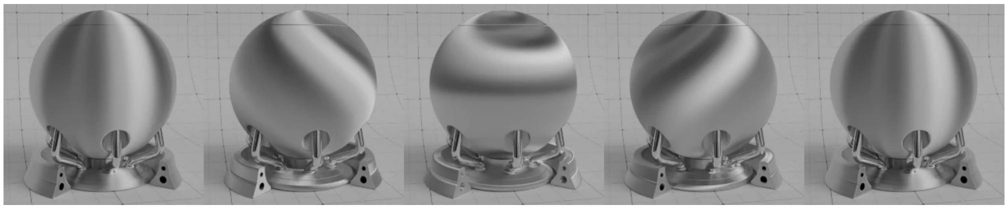


Anisotropic from 0.0 to 1.0

Anisotropic Rotation *Cycles Only*

Rotates the direction of anisotropy, with 1.0 going full circle.

Compared to the *Glossy BSDF* node, the direction of highlight elongation is rotated by 90°. Add 0.25 to the value to correct.



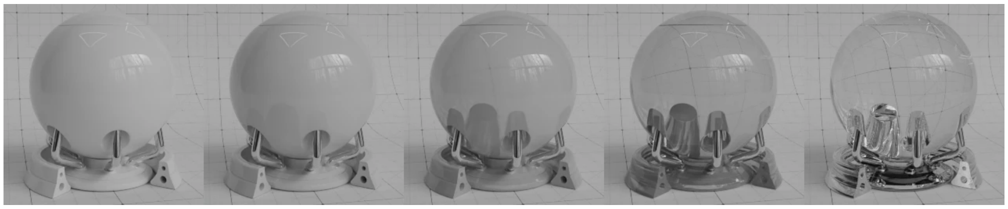
Anisotropic rotation from 0.0 to 1.0

Tangent
Controls the tangent direction for anisotropy.

Transmission

Transmission is used to render materials like glass and liquids, where the surface both reflects light and transmits it into the interior of the object

Weight
Mix between fully opaque surface at zero and fully transmissive at one.



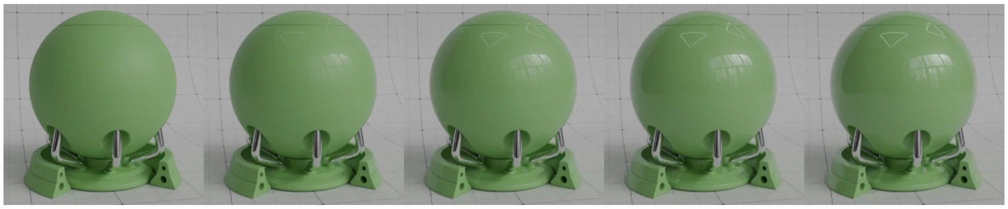
Weight from 0.0 to 1.0

Coat

Coat on top of the materials, to simulate for example a clearcoat, lacquer or car paint.

Weight
Controls the intensity of the coat layer, both the reflection and the tinting. Typically should be zero or one for physically-based materials,

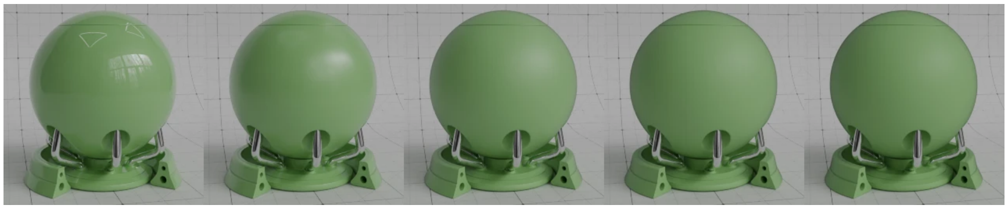
but may be textured to vary the amount of coating across the surface.



Weight from 0.0 to 1.0

Roughness

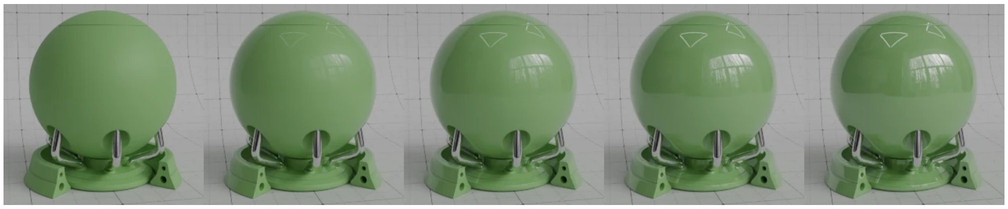
Roughness of the coat layer.



Roughness from 0.0 to 1.0

IOR

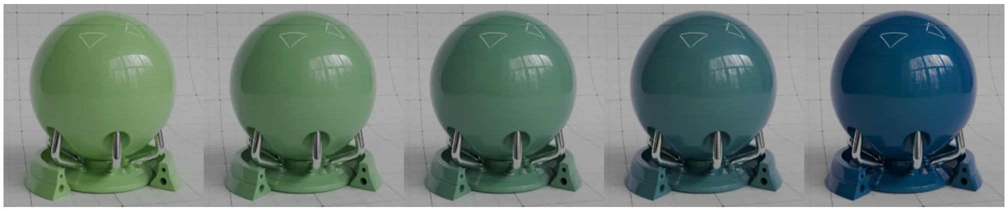
Index of refraction (IOR) of the coat layer. Affects its reflectivity as well as the falloff of coat tinting.



IOR from 1.0 to 2.0

Tint

Adds a colored tint to the coat layer by modeling absorption in the layer. Saturation increases at shallower angles, as the light travels farther through the medium, depending on the IOR.



Tint from white to blue

Normal

Controls the normals of the *Coat* layer, for example to add a smooth coating on a rough surface.

Sheen

Sheen simulates very small fibers on the surface. For cloth this adds a soft velvet like reflection near edges. It can also be used to simulate dust on arbitrary materials.

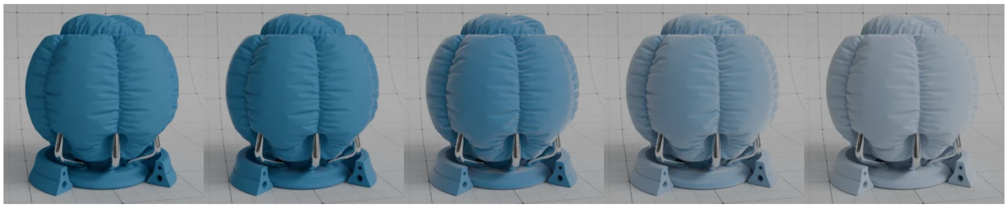
Weight

Controls in the intensity of the sheen layer.

Weight from 0.0 to 1.0

Roughness

Roughness of the sheen reflection.



Roughness from 0.0 to 1.0

Tint

The color of the sheen reflection.

Tint from white to green.

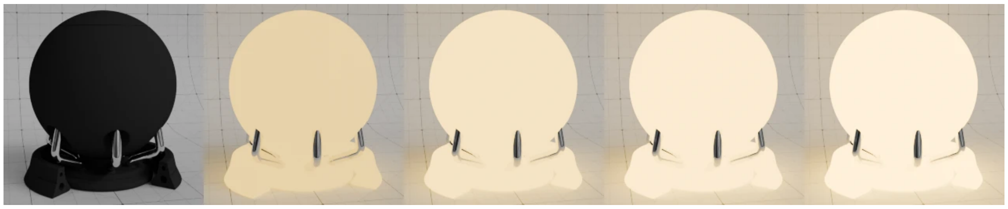
Emission

Light emission from the surface.

Color
: Color of light emission from the surface.

Emission color variations

Strength
: Strength of the emitted light. A value of 1.0 ensures that the object in the image has the exact same color as the *Emission Color*, i.e. make it ‘shadeless’.



Strength from 0.0 to 10.0

Thin Film *Cycles Only*

Thin Film simulates the effect of interference in a thin film sitting on top of the material. This causes the specular reflection to be colored in a way which strongly depends on the view angle as well as the film thickness and the index of refraction (IOR) of the film and the material itself.

This effect is commonly seen on e.g. oil films, soap bubbles or glass coatings. While its influence is more obvious in specular highlights, it also affects transmission.

Note

Thin-film interference is currently only applied to dielectric materials. Support for thin films on top of Metallic is planned in the future.

Thickness
: The thickness of the film in nanometers. A value of 0 disables the simulation. The interference effect is strongest between roughly 100 and 1000 nanometers, since this is near the wavelengths of visible light.

IOR
: Index of refraction (IOR) of the thin film. The common range for this value is between 1.0 (vacuum and air) and roughly 2.0, though some materials can reach higher values. The default value of 1.33 is a good approximation for water. Note that when the value is set to 1.0 or to the main IOR of the material, the thin film effect disappears since the film optically blends into the air or the material.

Outputs

BSDF
: Standard shader output.

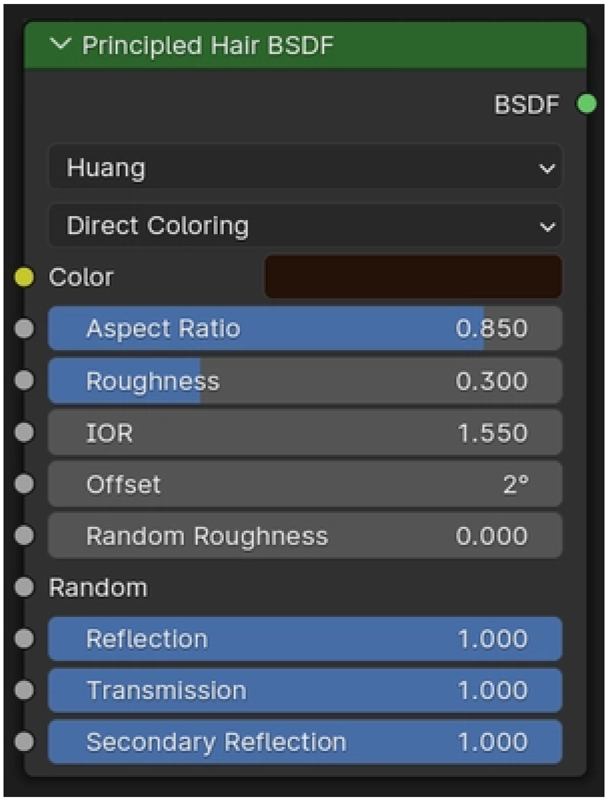
Principled Hair BSDF

Cycles Only

The *Principled Hair* BSDF is a physically-based, easy-to-use shader for rendering hair and fur.

Tip

Realistic hair should have a minimum of variance between each strand. The shader allows for this by specifying two values, *Random Color* and *Random Roughness*, which remap the specified Melanin/Roughness values to the range \(Color/Roughness \pm Randomization\%\).



Principled Hair BSDF

BSDF

Huang

Direct Coloring

Color

Aspect Ratio 0.850

Roughness 0.300

IOR 1.550

Offset 2°

Random Roughness 0.000

Random

Reflection 1.000

Transmission 1.000

Secondary Reflection 1.000

Inputs

Common

Color

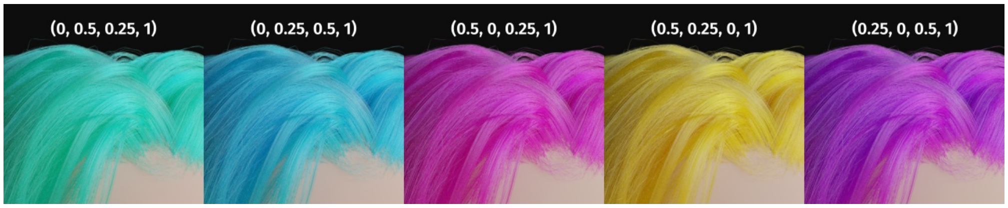
The RGB color of the strand. Only used in Direct coloring.

Hint

The chosen color is converted to an absorption coefficient with the following formula (section 4.2 of [CBTB16]):

$$\sigma_{a} = \frac{\ln(Color)} {\left(5.969 - 0.215\beta_{N} + 2.532\beta_{N}^{2} - 10.73\beta_{N}^{3} + 5.574\beta_{N}^{4} + 0.245\beta_{N}^{5}\right)^{2}}$$

where β_{N} is the radial roughness of the hair after applying randomization (if specified).



Coloring hair using the Direct coloring parametrization. (The numbers on top are the RGB values.)

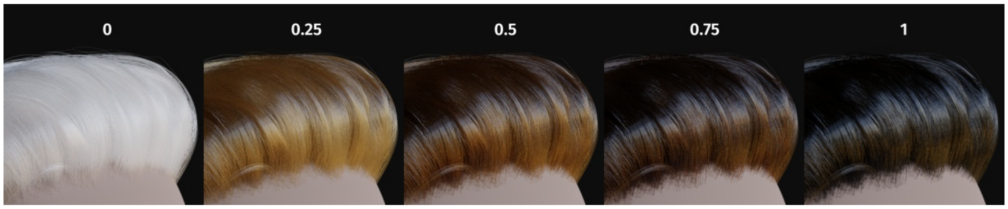
Melanin

Absolute quantity of pigment. Range $[0, 1]$ equivalent to $[0\%, 100\%]$.

Hint

This is a linear mapping to the underlying exponential function:

$$melanin\_qty = -\ln(\max(1.0 - Melanin, 0.0001))$$



Melanin.

Melanin Redness

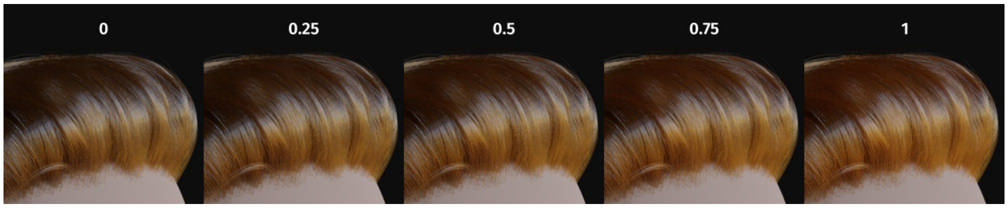
Ratio of pheomelanin to eumelanin. Range $[0, 1]$ equivalent to $[0\%, 100\%]$.

Hint

The ratio formula is: $eumelanin = Melanin \times (1.0 - MelaninRedness)$, $pheomelanin = Melanin \times MelaninRedness$.

The resulting quantities are converted (after randomization, if specified) to absorption concentration via the following formula (section 6.1 of [EFHLA11], adjusted for the range $[0, 1]$):

$$\begin{split}\sigma_{a} = eumelanin \times \left[\begin{matrix} 0.506 \\ 0.841 \\ 1.653 \\ \end{matrix}\right] + pheomelanin \times \left[\begin{matrix} 0.343 \\ 0.733 \\ 1.924 \\ \end{matrix}\right]\end{split}$$



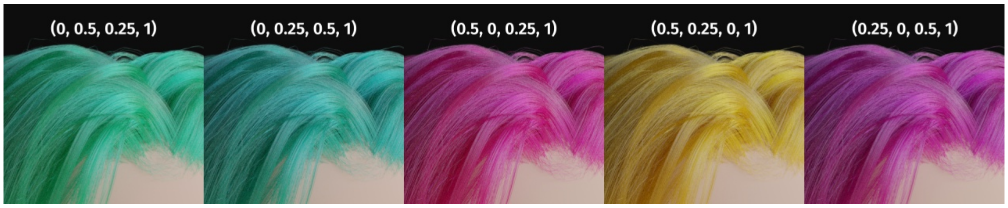
Melanin Redness.

Tint

Color used for dyeing the hair after applying the melanin pigment. It is not subject to randomization. It can be disabled by setting the color to white.

Hint

This is converted via the Color mapping above and added to the absorption coefficient of the melanin concentration.



Tint, using Melanin 0.1 and the corresponding RGB values.

Absorption Coefficient

Attenuation coefficient σ.

IOR

Index of refraction (IOR) defining how much the ray changes direction. At 1.0 rays pass straight through like in a transparent material; higher values give more refraction. Default value is 1.55.

Offset

Tilts the glint of the hair by increasing the angle of the scales of the hair’s cuticle with respect to the hair shaft. Human hair usually has low values.

Random Color

For each strand, vary the melanin concentration by $RandomFactor$. Range $[0, 1]$ equivalent to $[0\%, 100\%]$ of the initial melanin concentration.

Hint

The melanin concentration is multiplied by $randomFactor$, where $randomFactor = 1.0 + 2.0\times(Random - 0.5) \times RandomColor$.

0
0.25
0.5
0.75
1

Random Color.

Random Roughness

For each strand, vary both Roughness values by $RandomFactor$. Range $[0, 1]$ equivalent to $[0\%, 100\%]$ of the initial roughness values.

Hint

The applied formula is the same one as for *Random Color*.

0
0.25
0.5
0.75
1

Random Roughness.

Random

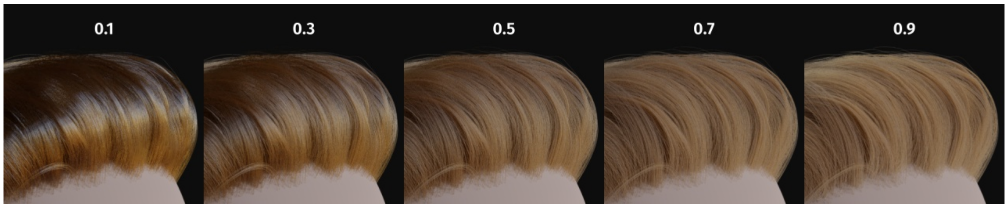
Random number source. If no node is connected here, it is automatically instanced with the value obtained from *Hair Info ▸ Random*.

Chiang Model

The Chiang model is based on a Gaussian distribution with separate roughness along and orthogonal to the hair.

Roughness

Specify how much the glints are smoothed in the direction of the hair shaft. Too low values will smoothen the hair to the point of looking almost metallic, making glints look like Fireflies; while setting it too high will result in a Lambertian look.



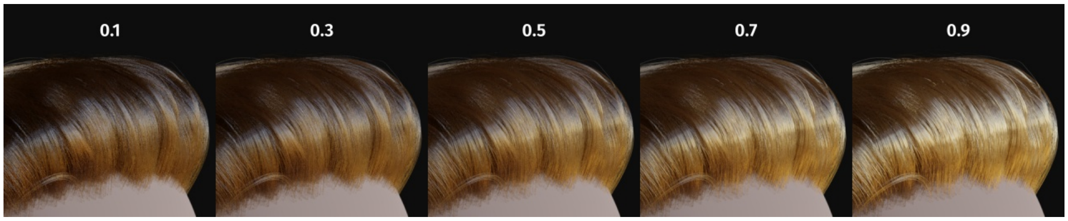
Roughness.

Radial Roughness

Specify how much the glints are smoothed in the direction of the hair normal. Too low values will concentrate the glint; while setting it too high will spread the light across the width of the strand.

Hint

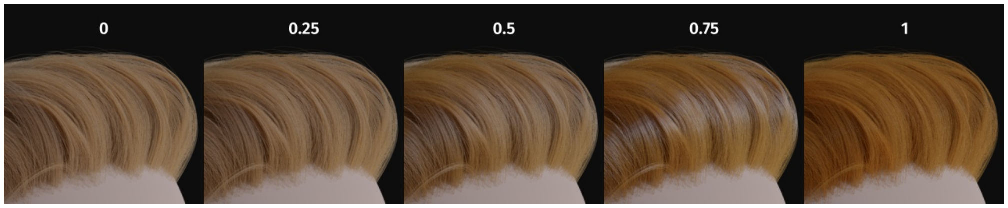
Mathematically, this parameter is mapped to the logistic distribution’s scale factor \(s\) (section 4.1 of [CBTB16]).



Radial Roughness.

Coat

Simulate a shiny coat of fur, by reducing the Roughness to the given factor only for the first light bounce (diffuse). Range $[0, 1]$ equivalent to a reduction of $[0\%, 100\%]$ of the original Roughness.



Coat.

Huang Model

The Huang model is based on microfacet based reflection and transmission, and supports elliptically shaped hair.

Aspect Ratio

The ratio of the minor axis to the major axis of an elliptical cross-section. Recommended values are 0.8~1 for Asian hair, 0.65~0.9 for Caucasian hair, 0.5~0.65 for African hair. The major axis is aligned with the curve normal, which can be created with geometry nodes, but is not supported in legacy particle hair.

Roughness

Microfacet roughness for reflection and transmission.

Reflection

Optional factor for modulating the first light bounce off the hair surface. The color of this component is always white. Keep this 1.0 for physical correctness.

Transmission

Optional factor for modulating the transmission component. Picks up the color of the pigment inside the hair. Keep this 1.0 for physical correctness.

Secondary Reflection

Optional factor for modulating the component which is transmitted into the hair, reflected off the backside of the hair and then transmitted out of the hair. This component is oriented approximately around the incoming direction, and picks up the color of the pigment inside the hair. Keep this 1.0 for physical correctness

Properties

Color Parametrization

The shader provides three different ways, or *parametrizations*, to color the hair strands.

Direct Coloring:: Choose the desired RGB color and the shader will approximate the necessary *absorption coefficient* (below).

Melanin Concentration::

This mode defines the color as the quantity and ratio of the pigments which are commonly found in hair and fur, *eumelanin* (prevalent in brown-black hair) and *pheomelanin* (red hair). The quantity is specified in the *Melanin* input, and the ratio between them in *Melanin Redness*. Increasing concentrations darken the hair (the following are with *Melanin Redness* \(1\)):

- White (Melanin \(0\))
- Blonde (Melanin \(0.25\))
- Reddish (Melanin \(0.5\))
- Brown (Melanin \(0.75\))
- Black (Melanin \(1\))

Additionally, the *Tint* inputs allows to dye the hair with the desired color.

Absorption Coefficient::

Specifies the attenuation coefficient \(\sigma_{a}\), as applied by the Beer-Lambert law. This mode is intended mainly for technical users who want to use coefficients from the literature without any sort of conversion.

Outputs

BSDF

Standard shader output.

References

This shader is an implementation of the papers by Chiang et al. [CBTB16] and Huang et al. [HHH22].

[CBTB16] (1,2,3)

Chiang, M. J. , Bitterli, B. , Tappan, C. and Burley, B. (2016), A Practical and Controllable Hair and Fur Model for Production Path Tracing. Computer Graphics Forum, 35: 275-283. doi:10.1111/cgf.12830

[EFHLA11]

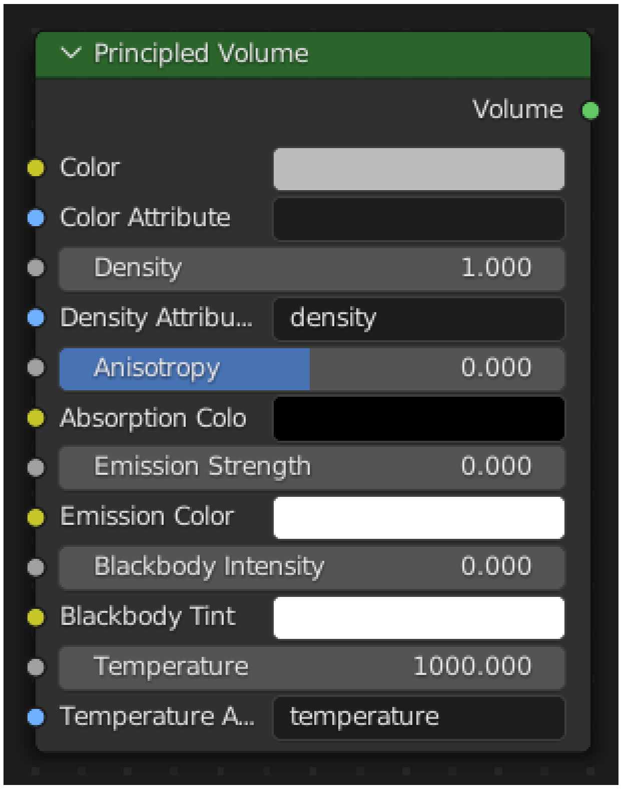
d'Eon, E. , Francois, G. , Hill, M. , Letteri, J. and Aubry, J. (2011), An Energy-Conserving Hair Reflectance Model. Computer Graphics Forum, 30: 1181-1187. doi:10.1111/j.1467-8659.2011.01976.x

[HHH22]

Huang W., Hullin M.B. Hanika J. (2022), A Microfacet-based Hair Scattering Model. Computer Graphics Forum, 41: 79-91. doi:10.1111/cgf.14588

Principled Volume

The *Principled Volume* shader combines all volume shading components into a single easy to use node. Volumes like smoke and fire can be rendered with a single shader node, which includes scattering, absorption and blackbody emission.



Color
Color Attribute
Density 1.000
Density Attribu... density
Anisotropy 0.000
Absorption Colo
Emission Strength 0.000
Emission Color
Blackbody Intensity 0.000
Blackbody Tint
Temperature 1000.000
Temperature A... temperature

Inputs

Color
: Volume scattering color.

Color Attribute
: Volume grid for coloring the volume. Use “color” for smoke simulations.

Density
: Density of the volume.

Density Attribute
: Volume grid to define the density, typically “density”.

Anisotropy
: Backward or forward scattering direction.

Absorption Color
: Volume shadow color tint.

Emission Strength
: Amount of light to emit.

Emission Color
: Emission color tint.

Blackbody Intensity
: Blackbody emission for fire. Set to 1 for physically accurate intensity.

Blackbody Tint
: Color tint for blackbody emission.

Temperature
: Temperature in kelvin for blackbody emission, higher values emit more.

Temperature Attribute
: Volume grid to define the temperature, typically "temperature".

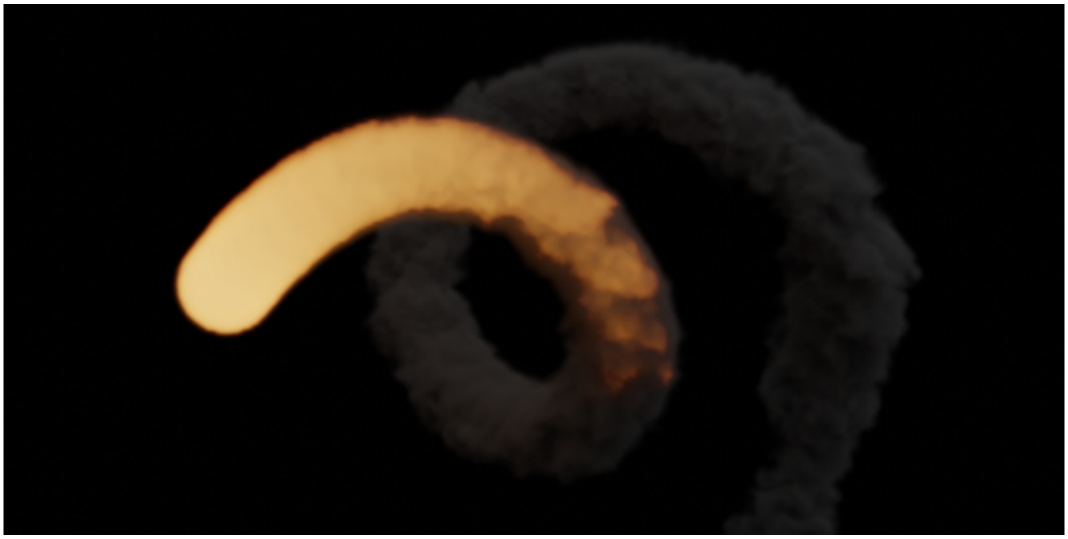
Properties

This node has no properties.

Outputs

Volume
: Standard shader output.

Examples



Ray Portal BSDF

Cycles Only

The *Ray Portal BSDF* node transports rays that enter to another location in the scene. It can be used to render portals for visual effects, and other production rendering tricks.

It acts much like a Transparent BSDF: render passes are passed through, and it is affected by light path max transparent bounces.

Note

- The *Ray Portal BSDF* only allows rays to pass through it in one direction. Add a second portal at the target location to make rays go in the other direction as well.
- Light sampling does not work efficiently through portals. This can lead to increased noise from lights on the other side of portals. Particularly small lights may be very noisy, or not pass through at all.

Inputs

Color
: Tint rays passing through the portal.

Position
: Ray start position at new location. Defaults to the current position, matching the Position output of the Geometry node.

Direction
: Ray direction at the new location. Defaults to the current view direction, which is the same as the negation of the Incoming output of the Geometry node.

Properties

This node has no properties.

Outputs

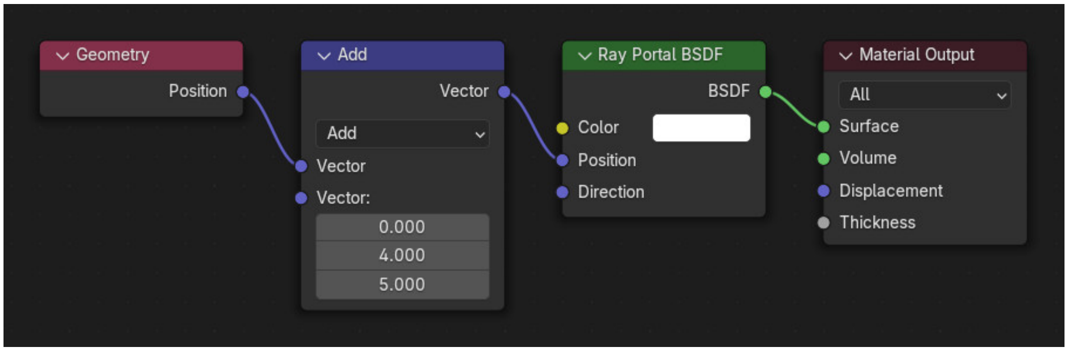
BSDF
    Standard shader output.

Examples

One use case for the *Ray Portal BSDF* is to connect two spaces together to create effects like a portal to an alternative dimension, or “impossible spaces” where something is bigger or smaller on the inside than expected.

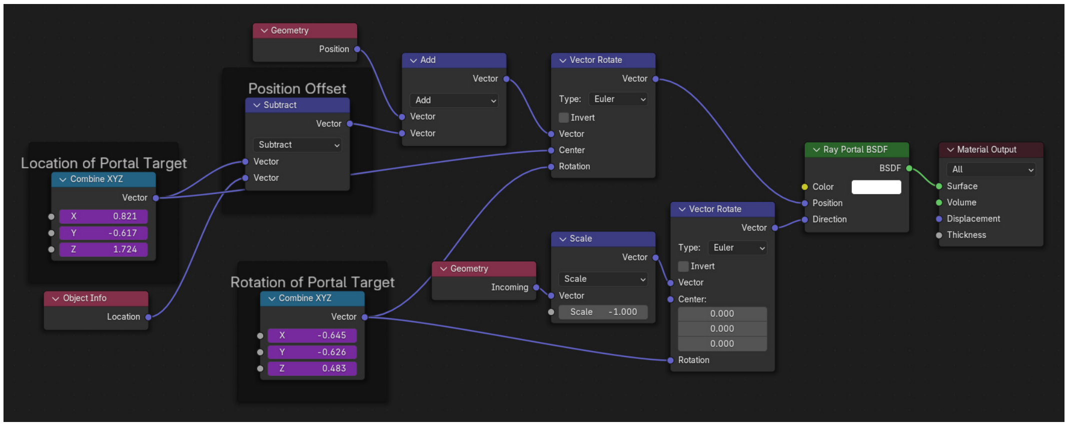
To set up a *Ray Portal BSDF* for a technique like this, augment the *Position* and *Incoming* outputs of the Geometry node to set the exit point and direction of the ray through the portal. Here are some examples:

Simple Offset


Geometry
Position
Add
Vector
Add
Vector
Vector:
0.000
4.000
5.000
Ray Portal BSDF
BSDF
Color
Position
Direction
Material Output
All
Surface
Volume
Displacement
Thickness

This simple node setup offsets the ray position. In this example, the ray is offset 0 units along the X axis, 4 units along the Y axis, and 5 units along the Z axis.

Portal

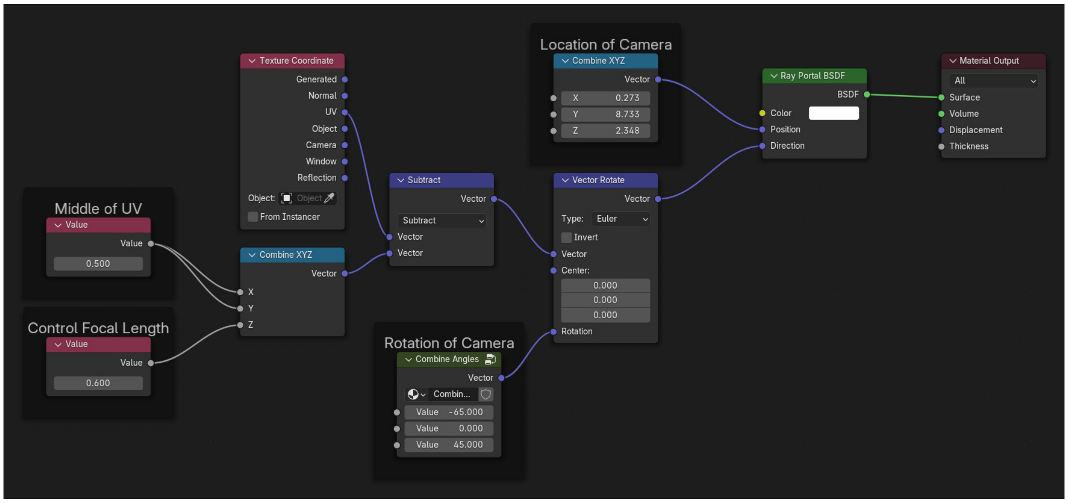

Geometry
Position
Add
Vector
Add
Vector
Vector
Vector Rotate
Vector
Type: Euler
Invert
Vector
Center
Rotation
Position Offset
Subtract
Vector
Subtract
Vector
Vector
Location of Portal Target
Combine XYZ
Vector
X 0.821
Y -0.617
Z 1.724
Object Info
Location
Ray Portal BSDF
BSDF
Color
Position
Direction
Material Output
All
Surface
Volume
Displacement
Thickness
Vector Rotate
Vector
Type: Euler
Invert
Vector
Center:
0.000
0.000
0.000
Rotation
Scale
Vector
Scale
Vector
Scale -1.000
Geometry
Incoming
Rotation of Portal Target
Combine XYZ
Vector
X -0.645
Y -0.626
Z 0.483

In this example, the *Location of Portal Target* and *Rotation of Portal Target* vectors are obtained from a target portal object using Drivers.

Camera Feed

Along with augmenting rays, the ray position and ray direction can be replaced, for effects like a camera feed on a screen.

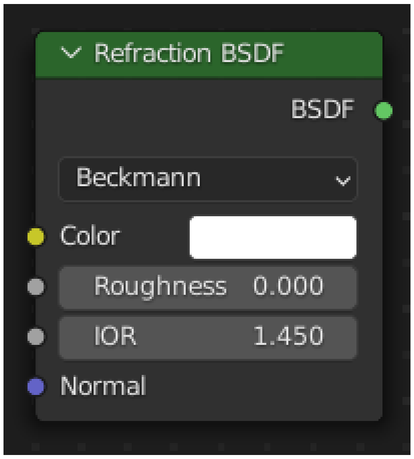
Using the Ray Portal BSDF to replicate the effect of a camera feed on a screen.


Location of Camera
Combine XYZ
Vector
X 0.273
Y 8.733
Z 2.348
Texture Coordinate
Generated
Normal
UV
Object
Camera
Window
Reflection
Object:
Object
From Instancer
Ray Portal BSDF
BSDF
Color
Position
Direction
Material Output
All
Surface
Volume
Displacement
Thickness
Middle of UV
Value
Value
0.500
Combine XYZ
Vector
X
Y
Z
Subtract
Vector
Subtract
Vector
Vector
Vector Rotate
Vector
Type: Euler
Invert
Vector
Center:
0.000
0.000
0.000
Rotation
Control Focal Length
Value
Value
0.600
Rotation of Camera
Combine Angles
Vector
Combin...
Value -65.000
Value 0.000
Value 45.000

Node setup for replicating a camera feed like effect on a screen.

Refraction BSDF

The *Refraction* BSDF is used to add glossy refraction with sharp or microfacet distribution, used for materials that transmit light. For best results this node should be considered as a building block and not be used on its own, but rather mixed with a glossy node using a Fresnel factor. Otherwise it will give quite dark results at the edges for glossy refraction.



Beckmann

Roughness 0.000

IOR 1.450

Inputs

Color

Color of the surface, or physically speaking, the probability that light is refracted for each wavelength.

Roughness

Influences sharpness of the refraction; perfectly sharp at 0.0 and smoother with higher values.

Normal

Normal used for shading; if nothing is connected the default shading normal is used.

Properties

Distribution

Microfacet distribution to use.

GGX::	GGX microfacet distribution.
Beckmann::	*Cycles Only* Beckmann microfacet distribution.

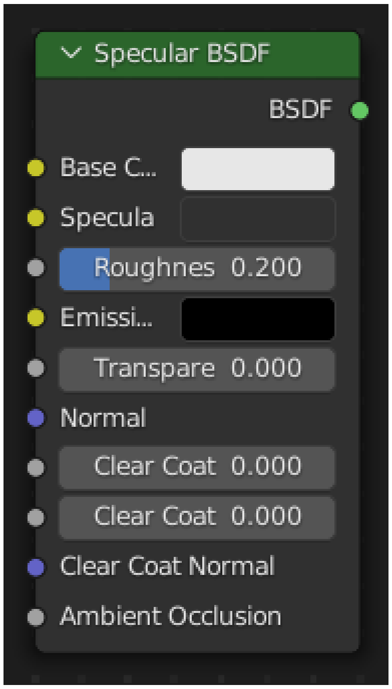
Outputs

BSDF
: Standard shader output.

Examples

Refraction Shader.

Specular BSDF


Specular BSDF
BSDF
Base C...
Specula
Roughnes 0.200
Emissi...
Transpare 0.000
Normal
Clear Coat 0.000
Clear Coat 0.000
Clear Coat Normal
Ambient Occlusion

EEVEE Only

The *Specular* BSDF combines multiple layers into a single easy to use node.

It is similar to the [Principled BSDF]() node but uses the *specular* workflow instead of the metallic. It has far fewer parameters and supports less features. Both might be merged into one node in the future.

The specular workflow functions by specifying the facing (along normal) reflection color. The result may not be physically plausible because there is no energy conservation.

Inputs

Base Color

Diffuse surface color. For conductor materials (metals) it should be black.

Specular

Amount of specular reflection. Specifies facing (along normal) reflectivity. Conductor materials (metals) can have colored specular reflection.

Hint

To compute this value for a realistic material with a known index of refraction, you may use this special case of the Fresnel formula: \(specular = ((ior - 1)/(ior + 1))^2\)

For example:

- water: ior = 1.33, specular = 0.02
- glass: ior = 1.5, specular = 0.04
- diamond: ior = 2.417, specular = 0.17

Roughness

Specifies microfacet roughness of the surface for diffuse and specular reflection.

Hint

When converting from the older *Glossy BSDF* node, use the square root of the original value.

Emissive Color

Color of the emitted light. This light is added to the BSDF result.

Transparency

Transparency factor. This is the inverse of the alpha channel (1 - alpha) you find in an image. Use an Invert node to convert alpha to transparency. This will only have an effect if the material uses a blend mode other than opaque.

Normal

Controls the normals of the base layers.

Clear Coat

Extra white specular layer on top of others. This is useful for materials like car paint and the like.

Clear Coat Roughness:

Roughness of clear coat specular.

Clear Coat Normal

Controls the normals of the *Clear Coat* layer.

Ambient Occlusion

Amount of occlusion to apply to indirect lighting. Usually a bake ambient occlusion map. The final occlusion factor is the minimum of this input and the runtime ambient occlusion effect.

Properties

This node has no properties.

Outputs

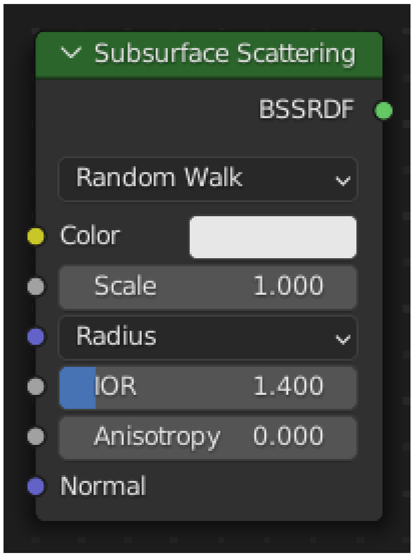
BSDF

Standard shader output.

Subsurface Scattering

The *Subsurface Scattering* node is used to add simple subsurface multiple scattering, for materials such as skin, wax, marble, milk and others. For these materials, rather than light being reflect directly off the surface, it will penetrate the surface and bounce around internally before getting absorbed or leaving the surface at a nearby point.

How far the color scatters on average can be configured per RGB color channel. For example, for skin, red colors scatter further, which gives distinctive red-colored shadows, and a soft appearance.



Random Walk
Scale 1.000
IOR 1.400
Anisotropy 0.000

Inputs

Color
: Color of the surface, or physically speaking, the probability that light is reflected for each wavelength.

Scale
: Global scale factor for the scattering radius.

Radius
: Average distance that light scatters below the surface. Higher radius gives a softer appearance, as light bleeds into shadows and through the object. The scattering distance is specified separately for the RGB channels, to render materials such as skin where red light scatters deeper. The X, Y and Z values are mapped to the R, G and B values, respectively.

IOR *Cycles Only*

Index of refraction for *Subsurface Scattering*.

Anisotropy *Cycles Only*
Directionality of subsurface scattering. Higher anisotropy scatters deeper into the object.

Roughness *Cycles Only*
Roughness of the glossy surface surrounding the subsurface volume.

Normal
Normal used for shading; if nothing is connected the default shading normal is used.

Properties

Subsurface Method
Rendering method to simulate subsurface scattering.

Note

EEVEE does not support the *Random Walk* methods.

Christensen-Burley::
An approximation to physically-based volume scattering. This method is less accurate than *Random Walk* however, in some situations this method will resolve noise faster.

Random Walk (Fixed Radius)::
Provides accurate results for thin and curved objects. Random Walk uses true volumetric scattering inside the mesh, which means that it works best for closed meshes. Overlapping faces and holes in the mesh can cause problems.

Random Walk::

Behaves similarly to *Random Walk (Fixed Radius)* but modulates the *Radius* based on the *Color*, *Anisotropy*, and *IOR*. This method thereby attempts to retain greater surface detail and color than *Random Walk (Fixed Radius)*.

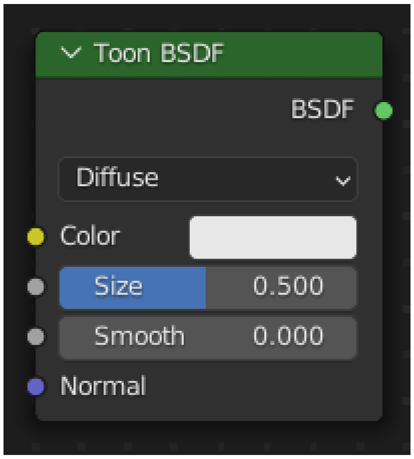
Outputs

BSSRDF
BSSRDF shader output.

Examples

Random walk subsurface scattering.

Toon BSDF



Cycles Only

The *Toon* BSDF is used to create *Diffuse* and *Glossy* materials with cartoon light effects.

Inputs

Color
: Color of the surface, or physically speaking, the probability that light is reflected for each wavelength.

Size
: Parameter between 0.0 and 1.0 that gives an angle of reflection between 0° and 90°.

Smooth
: This value specifies an angle over which a smooth transition from full to no reflection happens.

Normal
: Normal used for shading; if nothing is connected the default shading normal is used.

Properties

Component

The material component to base the toon effect.

Diffuse::	Use shading based on the Diffuse BSDF.
Glossy::	Use shading based on the Glossy BSDF for specular reflection.

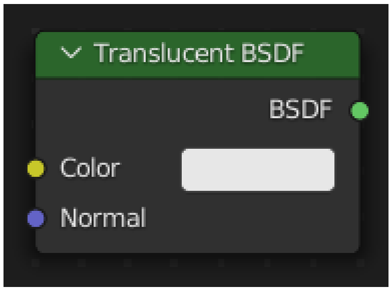
Outputs

BSDF
Standard shader output.

Examples

Example of Toon Shader.

Translucent BSDF



The *Translucent* BSDF is used to add Lambertian diffuse transmission.

Inputs

Color
: Color of the surface, or physically speaking, the probability that light is transmitted for each wavelength.

Normal
: Normal used for shading; if nothing is connected the default shading normal is used.

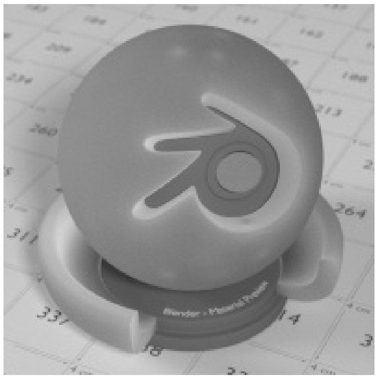
Properties

This node has no properties.

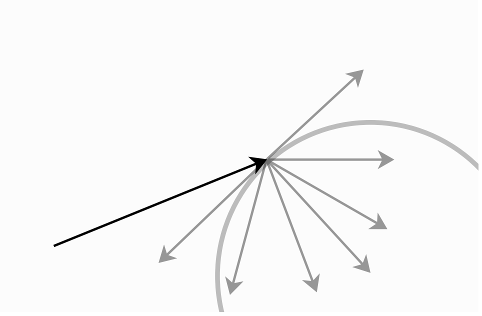
Outputs

BSDF
: Standard shader output.

Examples

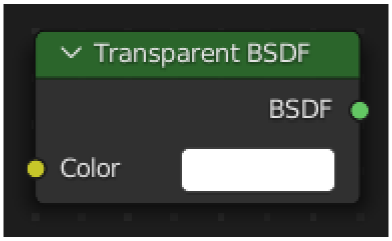


Translucent shader example.



Translucent shader behavior.

Transparent BSDF



The *Transparent* BSDF is used to add transparency without refraction, passing straight through the surface, as if there were no geometry there. Useful with alpha maps, for example. This shader affects light paths somewhat differently than other BSDFs. Note that only pure white transparent shaders are completely transparent.

Inputs

Color
: Color of the surface, or physically speaking, the probability for each wavelength that light is blocked or passes straight through the surface.

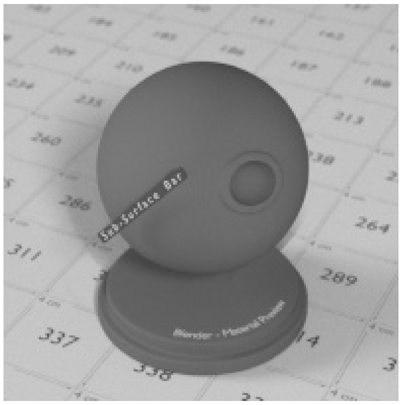
Properties

This node has no properties.

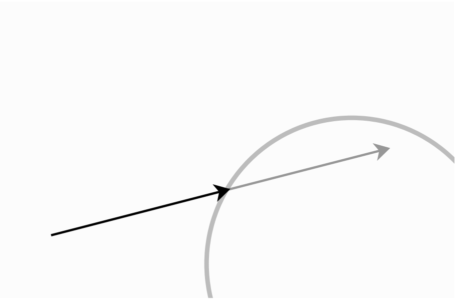
Outputs

BSDF
: Standard shader output.

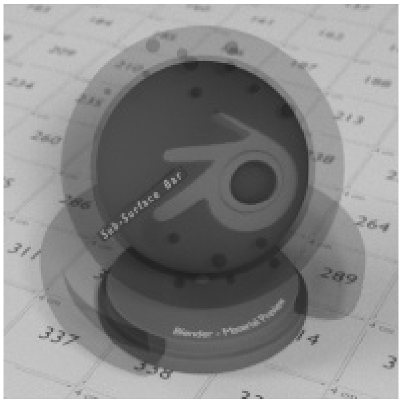
Examples



Transparent shader (pure white).



Transparent shader behavior.

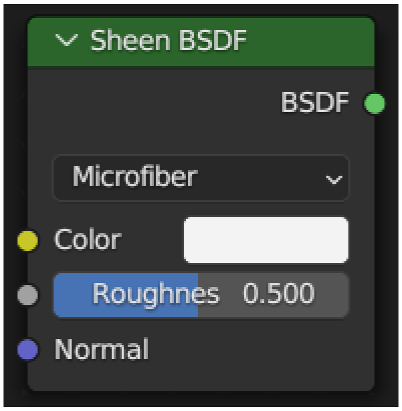


Transparent shader (gray).

Sheen BSDF

Cycles Only

The *Sheen* BSDF is used to add reflection to materials that have micro surface details such as cloth or dust. This shader is intended to be layered on top of other shaders such as dielectric or metallic shader setups.



Inputs

Color

Color of the surface, or physically speaking, the probability that light is reflected for each wavelength.

Roughness

Controls the amount of color that is reflected back to the camera, higher values reflect more color and can give a dusty appearance, while lower values look fuzzy and darker.

Normal

Normal used for shading; if nothing is connected the default shading normal is used.

Properties

Distribution

Sheen shading model.

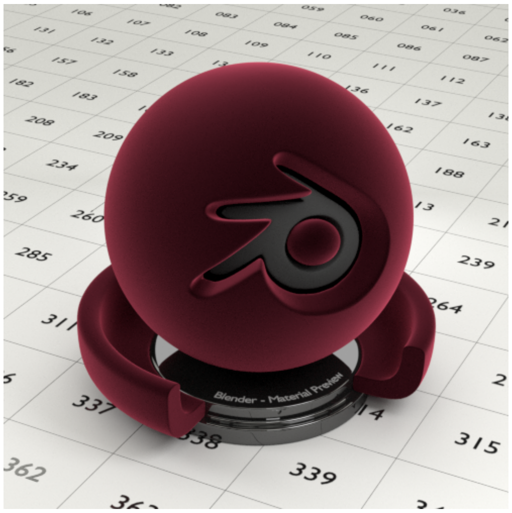
Ashikhmin:: Classic Ashikhmin velvet, used in Blender versions prior to 4.0

Microfiber::	Microflake-based model of multiple scattering between normal-oriented fibers.

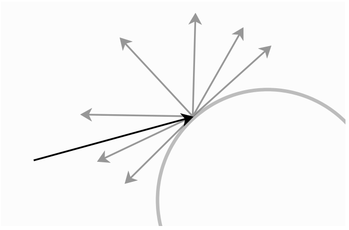
Outputs

BSDF
Standard shader output.

Examples

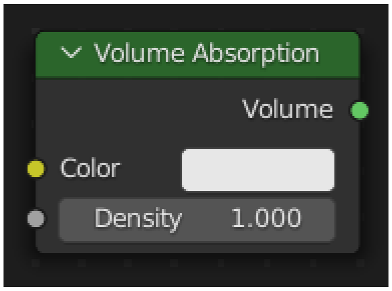


The Sheen shader example.



The Sheen shader behavior.

Volume Absorption


Volume Absorption
Volume
Color
Density 1.000

The *Volume Absorption* node allows light to be absorbed as it passes through the volume. Typical usage for this node would be water and colored glass.

Inputs

Color
: Color of the volume.

Density
: The density of the absorption effect.

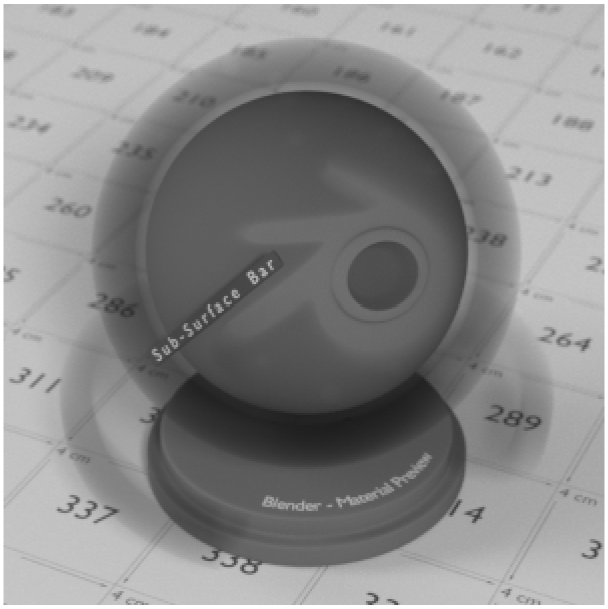
Properties

This node has no properties.

Outputs

Volume
: The Volume Shader output must be plugged into the *Volume Input* of the Material or World Output node.

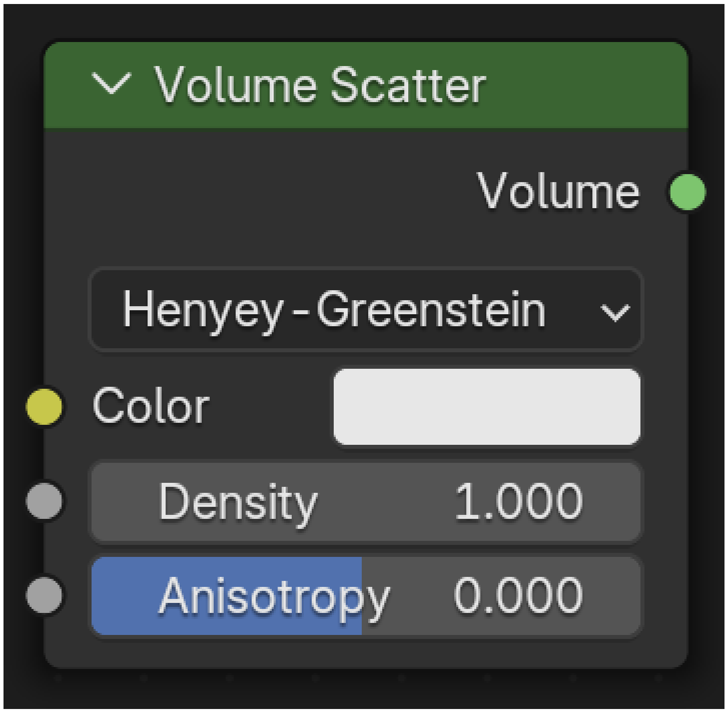
Examples


Sub-Surface Bar
Blender - Material Preview

Example of Volume Absorption.

Volume Scatter

The *Volume Scatter* node allows light to be scattered as it passes through the volume. Typical usage would be to add fog to a scene. It can also be used with the Volume Absorption node to create smoke.



Henyey-Greenstein

Density 1.000

Anisotropy 0.000

Inputs

Color
: Scattering coefficients per color channel.

Density
: The density of the scatter effect.

Anisotropy *Henyey-Greenstein Draine*
: Controls the relative amount of backward and forward scattering.

IOR *Fournier-Forand*
: Refractive index of the scattering particles relative to water. Common ocean waters range between 1.0 and 1.2, while turbid waters with higher density of particles have higher IORs.

Backscatter *Fournier-Forand*

Fraction of light that is scattered backwards. Most oceanic particles have backscatter values between 0.001 (e.g., very large phytoplankton) and 0.1 (e.g., very small mineral particles), pure water has a backscatter of 0.5. Values taken from Ocean Optics Web Book.

Alpha *Draine*

Blending factor between Henyey-Greenstein (\(\alpha = 0\)) and Cornette & Shanks (\(\alpha = 1\)) phase functions.

Diameter *Mie*

Diameter of the scattering particles in μm.

Properties

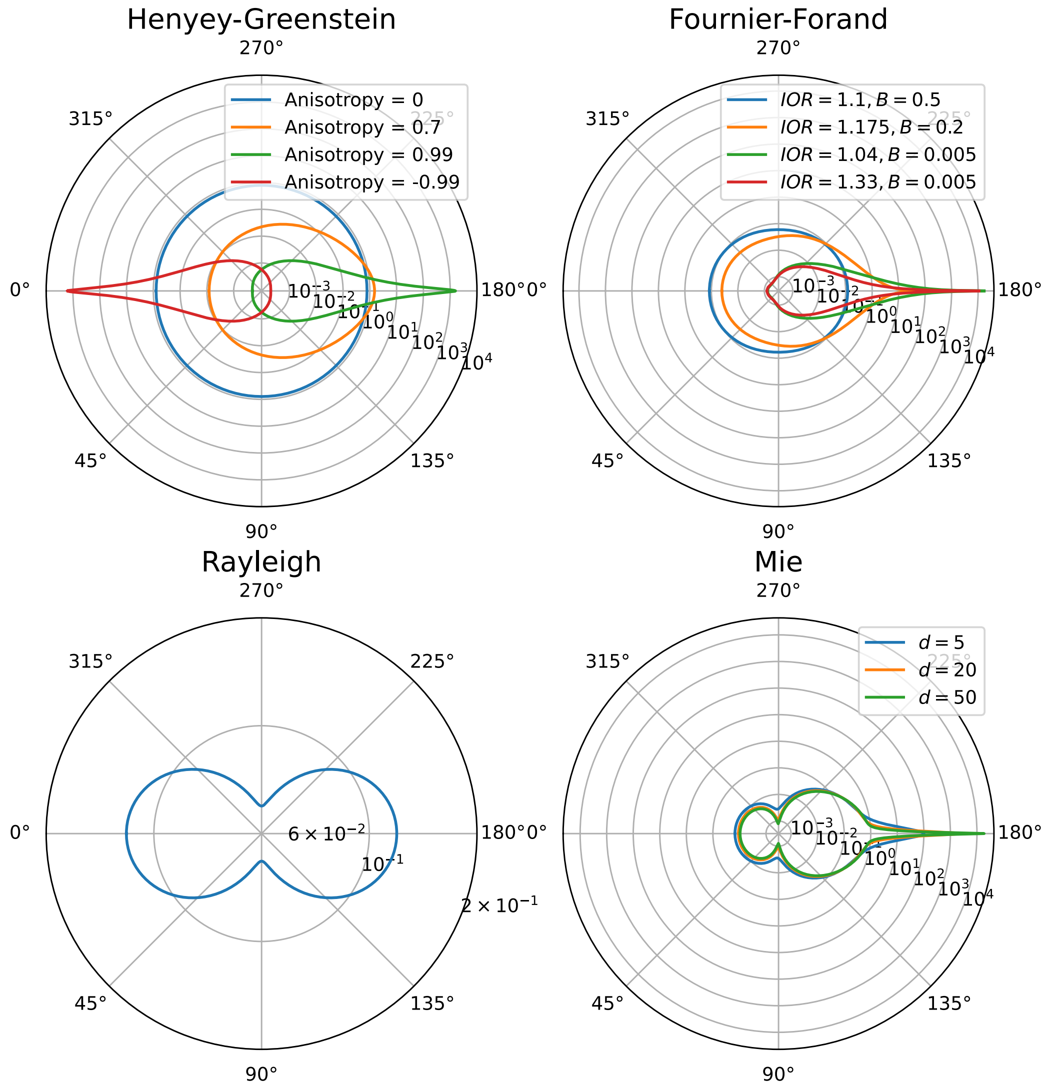
Phase

Volume scattering phase function.

Henyey-Greenstein::	Simple and widely used phase function, useful for approximating scattering in biological tissues.
Fournier-Forand::	*Cycles Only* Suitable for modeling the scattering of light in underwater environments.
Draine::	*Cycles Only* Suitable for modeling the scattering of interstellar dust.
Rayleigh::	*Cycles Only* Describes the scattering by particles with a size smaller than the wavelength of light, such as the scattering of sunlight in earth’s atmosphere.
Mie::	*Cycles Only* Describes the scattering by particles with a size larger than the wavelength of light, such as cloud and fog.

Tip

These phase functions can be combined using a Mix Shader.

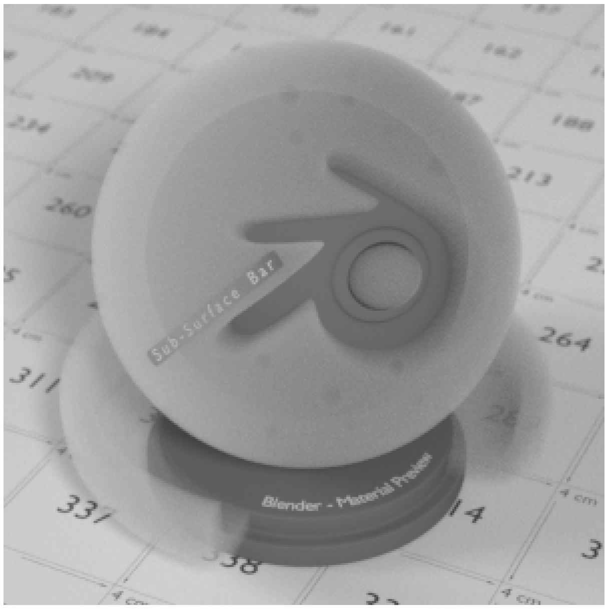


Volume scattering phase as a function of angles between the incoming and the outgoing direction, in logarithmic scale. Light comes from the left side.

Outputs

Volume
: The Volume Shader output must be plugged into the *Volume Input* of the Material or World Output node.

Examples


Sub-Surface Bar
Blender - Material Preview

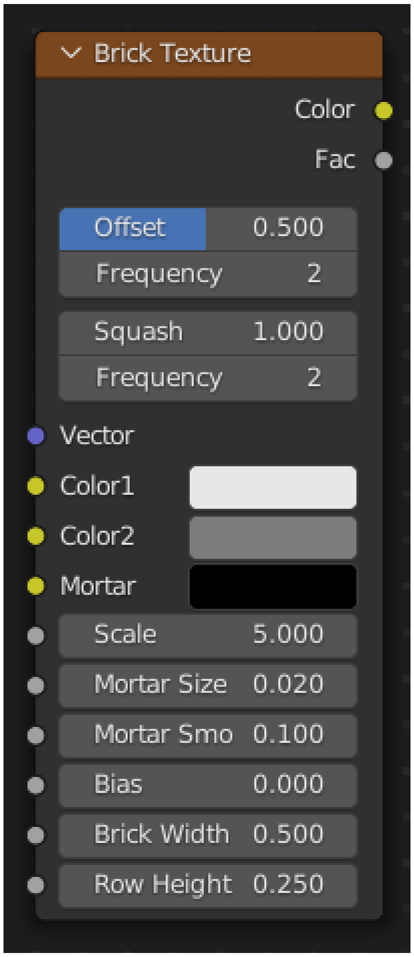
Example of Volume Scatter.

Texture

- Brick Texture Node
- Checker Texture Node
- Environment Texture Node
- Gabor Texture Node
- Gradient Texture Node
- IES Texture Node
- Image Texture Node
- Magic Texture Node
- Musgrave Texture Node
- Noise Texture Node
- Point Density Node
- Sky Texture Node
- Voronoi Texture Node
- Wave Texture Node
- White Noise Texture Node

Brick Texture Node

The *Brick Texture* is used to add a procedural texture producing bricks.


Brick Texture
Color
Fac
Vector
Color1
Color2
Mortar

Offset 0.500
Frequency 2
Squash 1.000
Frequency 2
Scale 5.000
Mortar Size 0.020
Mortar Smo 0.100
Bias 0.000
Brick Width 0.500
Row Height 0.250

Inputs

Color 1/2
: Color of the bricks.

Mortar
: The color of the area between bricks.

Scale
: Overall texture scale.

Mortar Size

The size of the filling between the bricks known as “mortar”; 0 means no mortar.

Mortar Smooth

Blurs/softens the edge between the mortar and the bricks. This can be useful with a texture and displacement textures.

Bias

The color variation between *Color 1/2*. Values of -1 and 1 only use one of the two colors; values in between mix the colors.

Brick Width

The ratio of brick’s width relative to the texture scale.

Row Height

The ratio of brick’s row height relative to the texture scale.

Properties

Offset

Determines the brick offset of the various rows.

Frequency

How often rows are offset; a value of 2 gives an even/uneven pattern of rows.

Squash

Factor to adjust the brick’s width for particular rows determined by the *Frequency*

Frequency

How often rows consist of “squished” bricks.

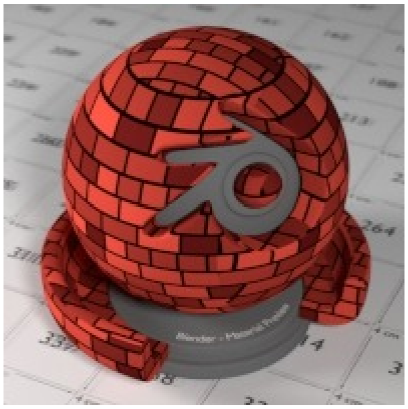
Outputs

Color

Texture color output.

Factor
Mortar mask (1 = mortar).

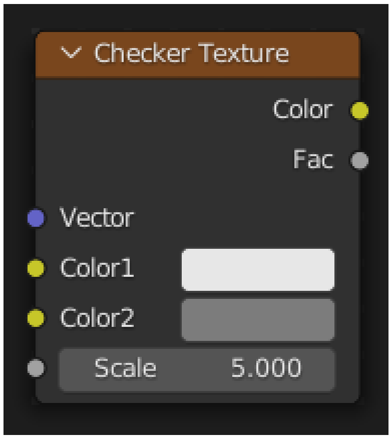
Examples



Brick texture: Colors changed, Squash 0.62, Squash Frequency 3.

Checker Texture Node

The *Checker Texture* is used to add a checkerboard texture.



Inputs

Vector
: Texture coordinate to sample texture at; defaults to Generated texture coordinates if the socket is left unconnected.

> **Warning**
>
> This node can have precision issues with some vector inputs. See the notes for the White Noise Texture for ways to mitigate this issue.

Color1, Color 2
: Color of the checkers.

Scale
: Overall texture scale. The scale is a factor of the bounding box of the face divided by the scale. For example, a scale of 15 will result in 15 alternate patterns over the overall UV bounding box. Different patterns could be achieved using other nodes to give different input patterns to this socket. For example, using the Math Node.

Properties

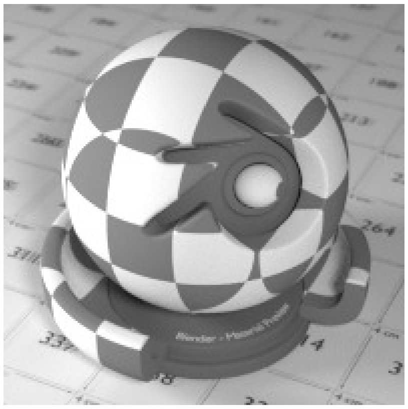
This node has no properties.

Outputs

Color
: Texture color output.

Factor
: Checker 1 mask (1 = Checker 1).

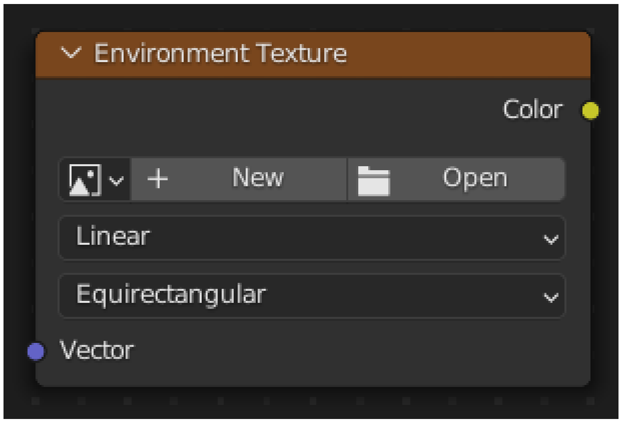
Examples



Default Checker texture.

Environment Texture Node

The Node *Environmental Texture* is used to light your scene using an environment map image file as a texture.



Inputs

Vector
Texture coordinate for texture look-up. If this socket is left unconnected, the image is mapped as environment with the Z axis as up.

Properties

Image
Image data-block used as the image source. Additional settings can be found in *Sidebar* ‣ *Item* ‣ *Properties*: These include options to control the alpha channel along with addition options for the color space. These addition options are documented with the rest of Common Image Settings.

Color Space
Type of data that the image contains, either Color or Non-Color Data. For most color textures the default of Color should be used, but in case of e.g. a bump or alpha map, the pixel values should be interpreted as Non-Color Data, to avoid doing any unwanted color space conversions.

The list of color spaces depends on the active OCIO config. The default supported color spaces are described in detail here: Default OpenColorIO Configuration

Texture Interpolation

Interpolation method used for the environment texture. The following interpolations are available:

Linear::	Regular quality interpolation.
Closest::	No interpolation, use closest pixel.
Cubic::	Smoother, better quality interpolation.
Smart::	Bicubic when magnifying, otherwise Bilinear is used. This is only available for OSL.

Projection Method

Allows you to use different types of environmental maps. The following methods are supported:

Equirectangular::	Projection from an Equirectangular photo.
Mirror Ball::	Projection from an orthographic photo or mirror ball.

Outputs

Color

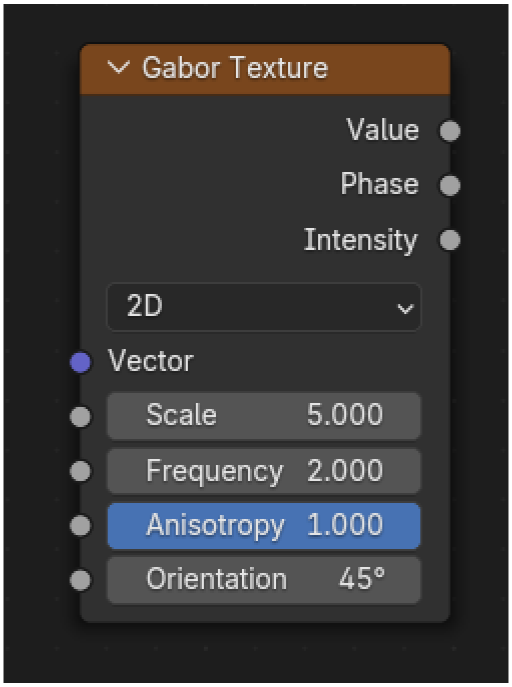
RGB color from the image.

Examples

HDR image from OpenFootage.net.

Gabor Texture Node

The *Gabor Texture* node evaluates a Gabor noise at the input texture coordinates. Gabor noise is visually characterized by random interleaved bands whose direction and width can be controlled. Additionally, it can be used to create omnidirectional noise like the standard Noise Texture node, but since it is more expensive to compute, using the Noise Texture node is probably the better option in those cases. See the examples for more information.



2D

Scale 5.000

Frequency 2.000

Anisotropy 1.000

Orientation 45°

Inputs

Vector

The coordinates at which Gabor noise will be evaluated. The Z component is ignored in the 2D case. Defaults to *Generated* texture coordinates if the socket is left unconnected.

Scale

Scale of the Gabor noise.

Frequency

The rate at which the Gabor noise changes across space. This is different from the Scale input in that it only scales perpendicular to the Gabor noise direction.

Anisotropy

The directionality of Gabor noise. 1 means the noise is completely directional, while 0 means the noise is omnidirectional.

Orientation

The direction of anisotropic Gabor noise. This is an angle for the 2D case, while it is a unit direction vector for the 3D case.

Properties

Type

Type of Gabor noise texture.

2D::	Evaluates the noise in 2D space. The Z component of the input vector is ignored.
3D::	Evaluates the noise in 3D space.

> **Note**
>
> Higher dimensions corresponds to higher render time, so lower dimensions should be used unless higher dimensions are necessary.

Outputs

Value

The Gabor noise value with both random intensity and phase. This is equal to sine the phase multiplied by the intensity.

Phase

The phase of the Gabor noise, which has no random intensity.

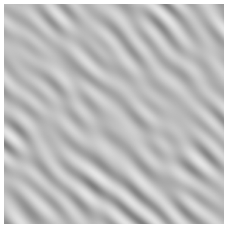
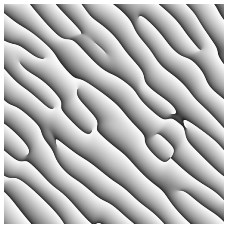
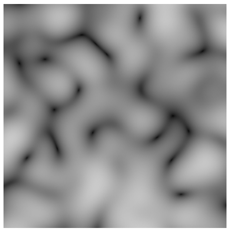
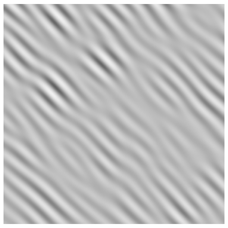
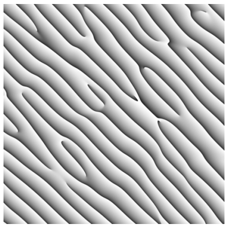
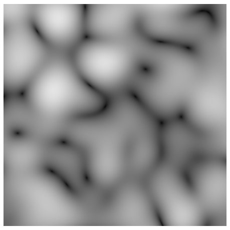
Intensity

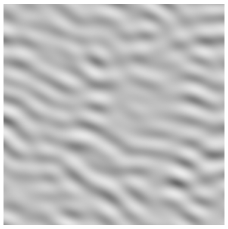
The intensity of the Gabor noise, which has no random phase.

Examples

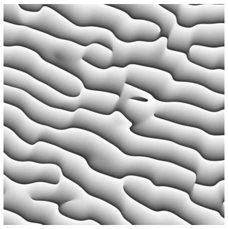
The following table demonstrates different outputs of the node with different parameters. As can be seen, the noise is visually characterized by interleaved bands that are generally oriented in a specific direction. But the *Anisotropy* parameter can be decreased below 1 to make the bands more random in directions. The *Frequency* parameter determines the number of bands perpendicular to the direction of the noise. However, the *Scale* parameter can also be used to globally increase the number of bands, so consider increasing the scale first since high frequency noise can suffer from low contrast and limited interleaving of bands.

Different outputs with different parameters.

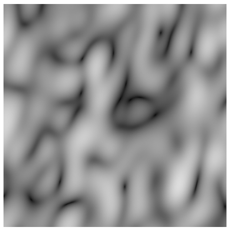
 Value output. Frequency = 2. Anisotropy = 1.	 Phase output. Frequency = 2. Anisotropy = 1.	 Intensity output. Frequency = 2. Anisotropy = 1.
 Value output. Frequency = 3. Anisotropy = 1.	 Phase output. Frequency = 3. Anisotropy = 1.	 Intensity output. Frequency = 3. Anisotropy = 1.



Value output. Frequency = 2. Anisotropy = 0.7.

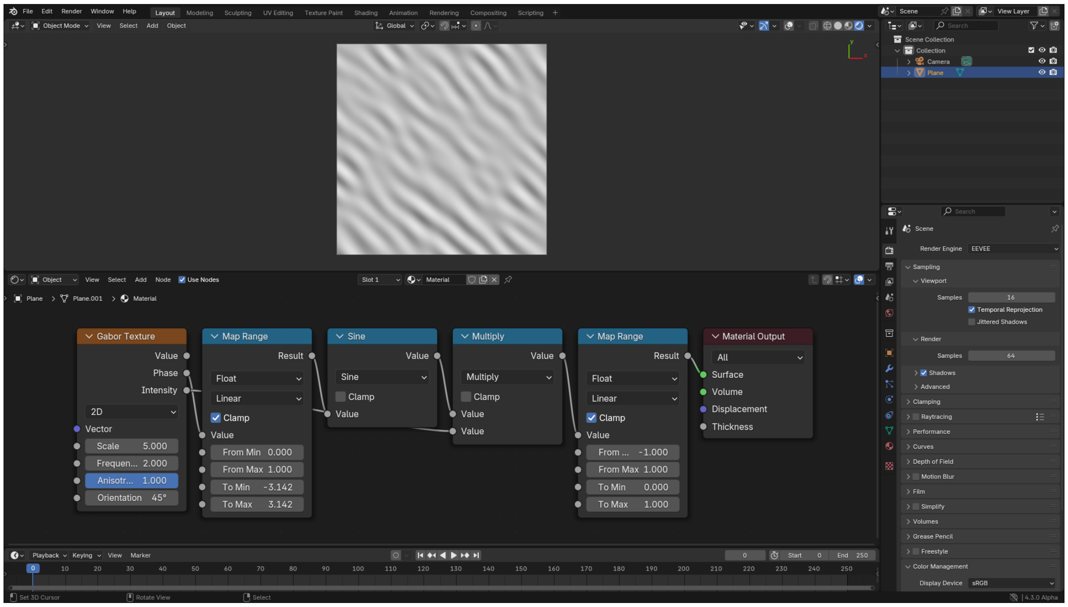


Phase output. Frequency = 2. Anisotropy = 0.7.



Intensity output. Frequency = 2. Anisotropy = 0.7.

Gabor noise is decomposed into a *Phase* and an *Intensity* components, where the Gabor value is computed as sine the phase multiplied by the intensity, noting that the phase output is normalized to the [0, 1] range.



Gabor Texture: 2D, Scale 5.000, Frequen... 2.000, Anisotr... 1.000, Orientation 45°

Map Range: Float, Linear, Clamp ☑, From Min 0.000, From Max 1.000, To Min -3.142, To Max 3.142

Sine: Sine, Clamp ☐

Multiply: Multiply, Clamp ☐

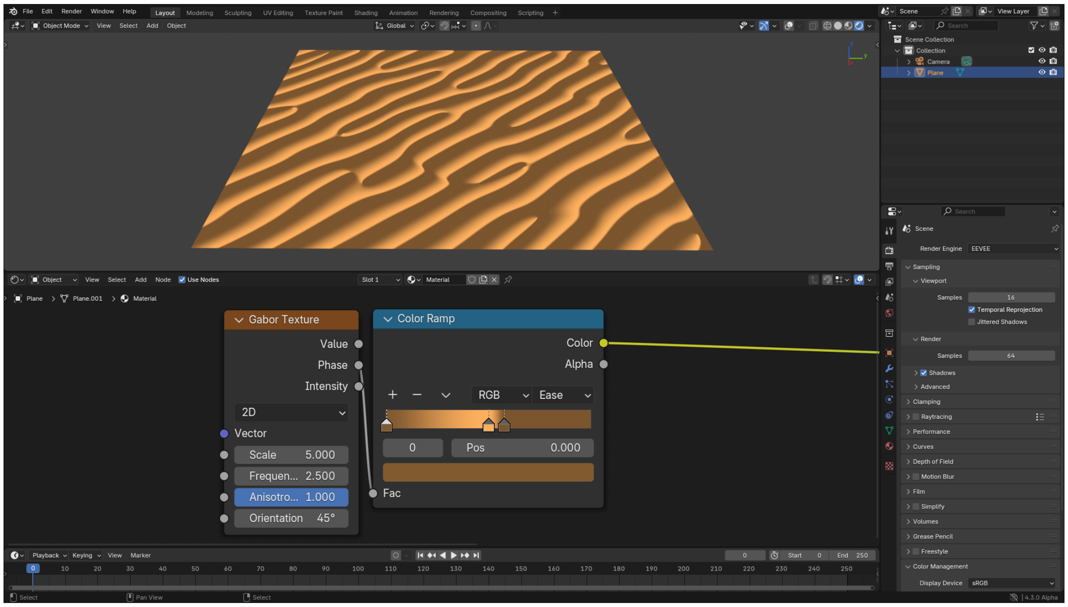
Map Range: Float, Linear, Clamp ☑, From ... -1.000, From Max 1.000, To Min 0.000, To Max 1.000

Material Output: All

Gabor Texture
Value
Phase
Intensity
Vector
Map Range
Result
Value
Sine
Value
Multiply
Value
Map Range
Result
Value
Material Output
Surface
Volume
Displacement
Thickness
Object
View
Select
Add
Node
Use Nodes
Slot 1
Material
Plane
Plane.001
Playback
Keying
Marker
Start
End
Rotate View

Compute the value output from the phase and intensity outputs.

The advantage of the *Phase* output is that it has no random intensities and no low contrast areas as in the value output, so it can be used as a base for textures that are more structured in nature, like sand dunes.


Gabor Texture
Value
Phase
Intensity
Vector
Color Ramp
Color
Alpha
Fac
Scene
Sampling
Viewport
Render
Shadows
Advanced
Color Management

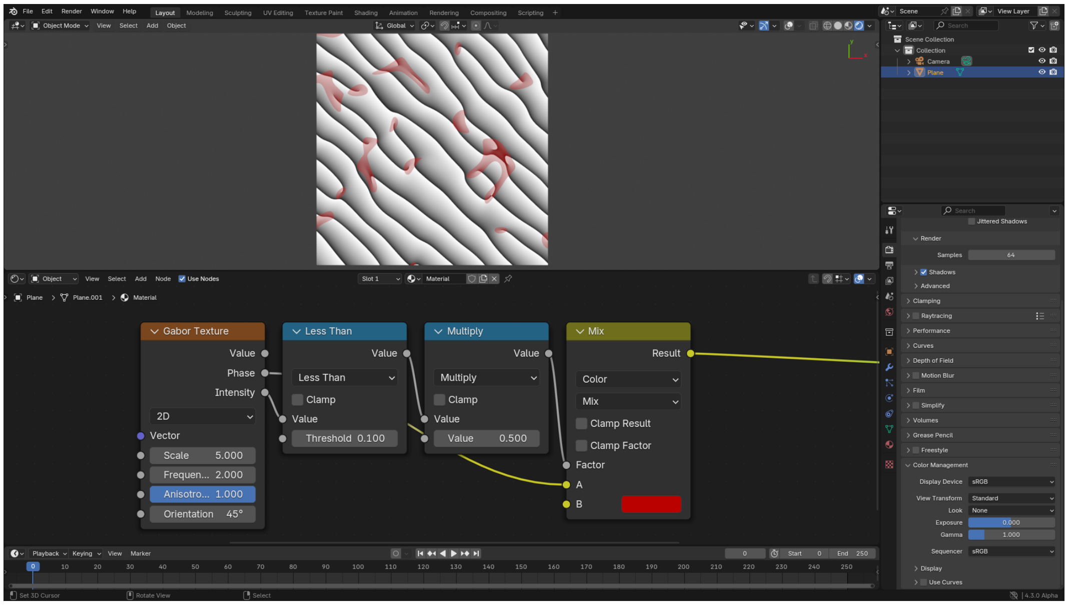
Gabor Texture: 2D
Scale 5.000
Frequen... 2.500
Anisotro... 1.000
Orientation 45°

Color Ramp: RGB, Ease
0 Pos 0.000

Render Engine EEVEE
Viewport Samples 16
☑ Temporal Reprojection
☐ Jittered Shadows
Render Samples 64
Display Device sRGB

Sand dune-like structures creates using the phase output.

The main advantage and use of the *Intensity* output is that it provides information about the location of *singularities* in the *Phase* output. Singularities are those areas in the phase where the bands meet, which are shown in red in the following figure. Those areas will be close to zero in the *Intensity* output. So if those areas are undesirable, they can be hidden by multiplying by a variant of the *Intensity* output.


Gabor Texture
Value
Phase
Intensity
Vector
Less Than
Value
Multiply
Value
Mix
Result
Factor
A
B

Gabor Texture: 2D
Scale 5.000
Frequen... 2.000
Anisotro... 1.000
Orientation 45°

Less Than: Less Than
☐ Clamp
Threshold 0.100

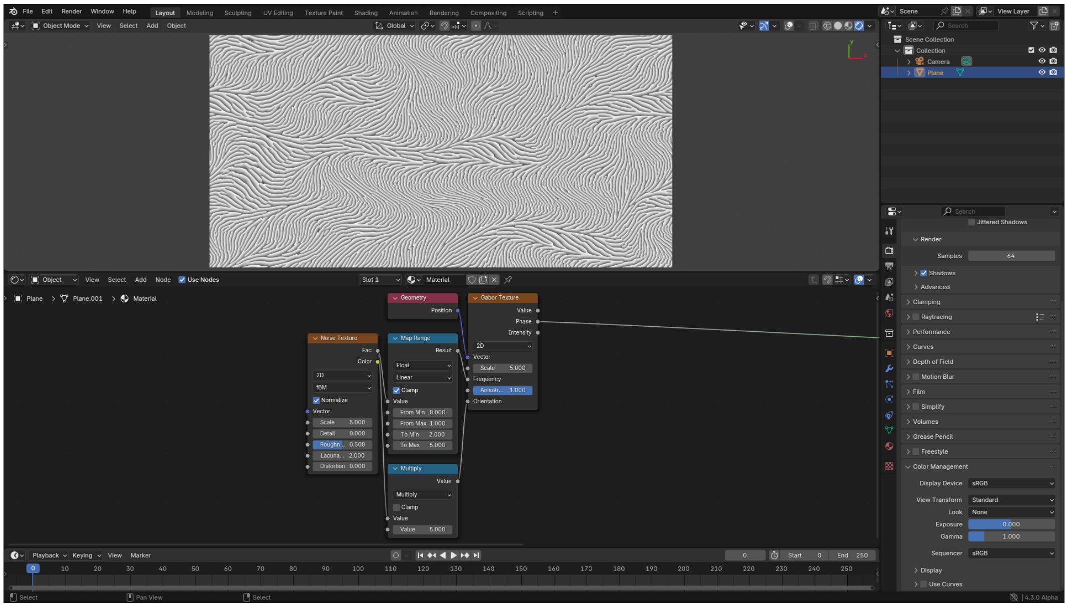
Multiply: Multiply
☐ Clamp
Value 0.500

Mix: Color, Mix
☐ Clamp Result
☐ Clamp Factor

View Transform Standard
Look None
Exposure 0.000
Gamma 1.000
Sequencer sRGB

Visualization of the areas where singularities happen.

Inputs can be varies across space to get more interesting patterns.


File
Edit
Render
Window
Help
Layout
Modeling
Sculpting
UV Editing
Texture Paint
Shading
Animation
Rendering
Compositing
Scripting
Object Mode
View
Select
Add
Object
Global
Scene
View Layer
Scene Collection
Collection
Camera
Plane
Object
Node
Use Nodes
Slot 1
Material
Plane.001
Playback
Keying
Marker
Start 0
End 250
Select
Pan View
4.3.0 Alpha

Geometry
- Position

Noise Texture
- Fac
- Color
- 2D
- fBM
- ☑ Normalize
- Vector
- Scale 5.000
- Detail 0.000
- Roughn... 0.500
- Lacuna... 2.000
- Distortion 0.000

Map Range
- Result
- Float
- Linear
- ☑ Clamp
- Value
- From Min 0.000
- From Max 1.000
- To Min 2.000
- To Max 5.000

Multiply
- Value
- Multiply
- ☐ Clamp
- Value
- Value 5.000

Gabor Texture
- Value
- Phase
- Intensity
- 2D
- Vector
- Scale 5.000
- Frequency
- Anisotr... 1.000
- Orientation

Jittered Shadows

Render
- Samples 64
- ☑ Shadows
- Advanced

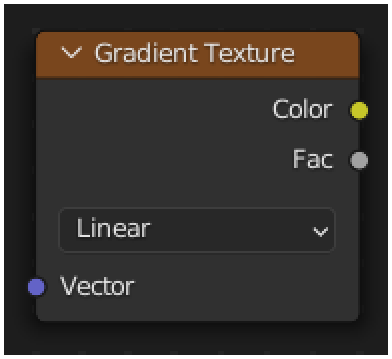
Clamping
Raytracing
Performance
Curves
Depth of Field
Motion Blur
Film
Simplify
Volumes
Grease Pencil
Freestyle

Color Management
- Display Device sRGB
- View Transform Standard
- Look None
- Exposure 0.000
- Gamma 1.000
- Sequencer sRGB
- Display
- Use Curves

Varying the frequency and orientation across space.

Gradient Texture Node

The *Gradient Texture* node generates interpolated color and intensity values based on the input vector.



Inputs

Vector

Texture coordinate to sample texture at; defaults to Generated texture coordinates if the socket is left unconnected.

Properties

Gradient Type

Controls the type of gradient generated.

Linear::	Directly outputs the input X coordinate.
Quadratic::	Interpolates the input X coordinate quadratically.
Easing::	Uses a combination of quadratic and linear interpolation to generate a smooth gradient from the input X coordinate.
Diagonal::	Averages the input X and Y coordinates.
Spherical::	Creates an inverse gradient using the length of the input vector; the maximum value is at (0, 0, 0).
Quadratic Sphere::	The same as Spherical, except interpolated quadratically.
Radial::	

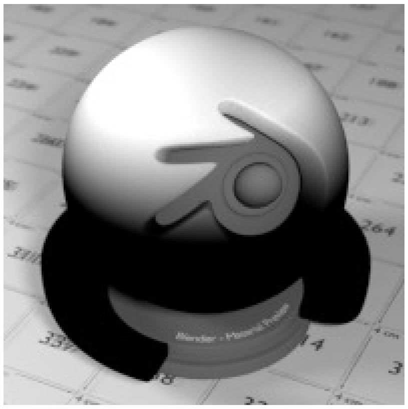
Outputs a value based on the angle of the input around the Z axis.

Outputs

Color
: Texture color output.

Factor
: Texture intensity output.

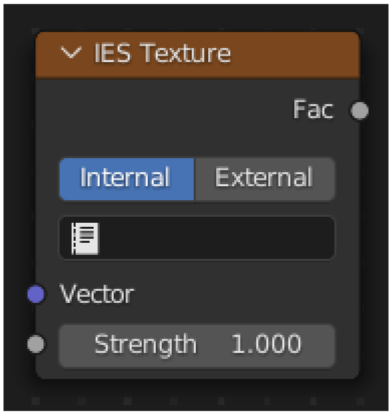
Examples



Gradient texture using object coordinates.

IES Texture Node

The *IES Texture* is used to match real world lights based on IES files (IES). IES files store the directional intensity distribution of light sources.



Inputs

Vector
: Texture coordinate for lookup in the light distribution. Defaults to the normal.

Strength
: Light strength multiplier.

Properties

Mode
: The location to load the IES file from.

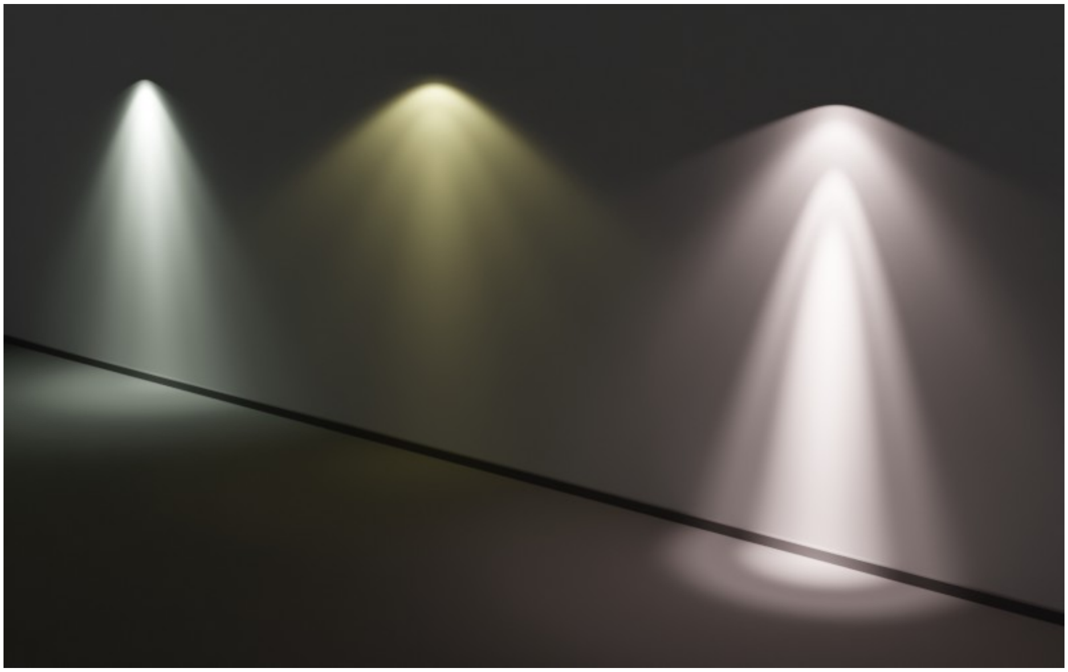
Internal::	Use IES profile from a file embedded in a text data-block in the blend-file, for easy distribution.
External::	Load IES profile from a file on the drive.

Outputs

Factor

Light intensity, typically plugged into the Strength input of an Emission node.

Examples



Lights with different IES profiles.

Image Texture Node

Used for applying an image as a texture.

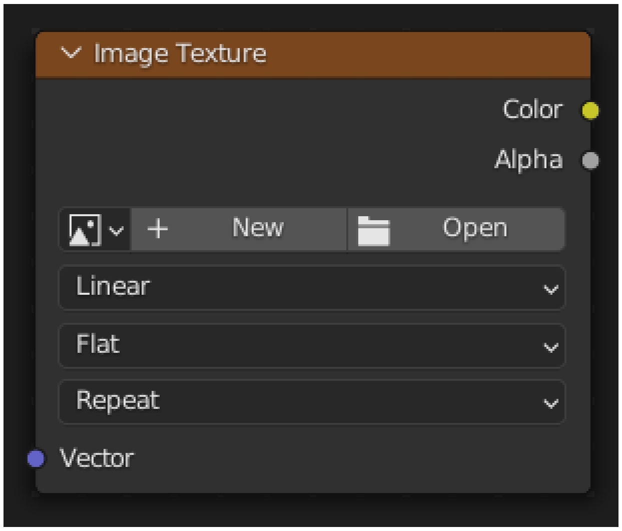


Image Texture
Color
Alpha
New
Open
Linear
Flat
Repeat
Vector

Inputs

Vector
: 3D coordinate that's projected onto the 2D image using the selected *Projection* method. The node then outputs the color and alpha at this projected point.

  This slot is usually connected to an output of the Texture Coordinate Node. If left unconnected, the coordinate is taken from the object's active UV map (with Z = 0).

Properties

Image
: Image data-block to use.

Interpolation
: Method to scale images up or down for rendering.

Linear::	Regular quality interpolation.
Cubic::	Smoother, better quality interpolation. Bump maps should use this for best results.
Closest::	No interpolation (nearest neighbor). Useful for rendering pixel art.
Smart::	*Cycles Only* Only for Open Shading Language. Use cubic interpolation when scaling up and linear when scaling down, for better performance and sharpness.

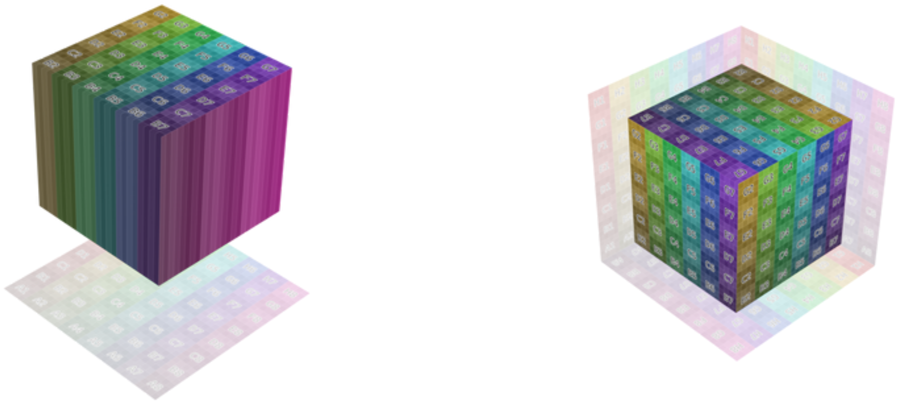
Projection

How to project *Vector* onto the image for arriving at a color.

Flat::	Place the image in a unit square (stretching from $(0, 0, 0)$ to $(1, 1, 0)$) and project the *Vector* vertically onto it. This projection is typically used in combination with UV maps.
Box::	Place the image on each side of a unit cube (stretching from $(0, 0, 0)$ to $(1, 1, 1)$) and project the *Vector* onto this cube, along the axis that's closest to the mesh normal. This projection is commonly used in architectural models considering these have lots of box-shaped objects. Blend Rather than projecting onto just one side (which creates sharp transitions), project onto multiple sides and blend the results together. The higher the value, the more blending and the smoother the result.
Sphere::	Wrap the image around a sphere with origin (0.5, 0.5, 0.5), and project the *Vector* from this origin onto this sphere. This projection is, of course, perfect for spherical objects such as planets, and is also useful for organic objects.
Tube::	

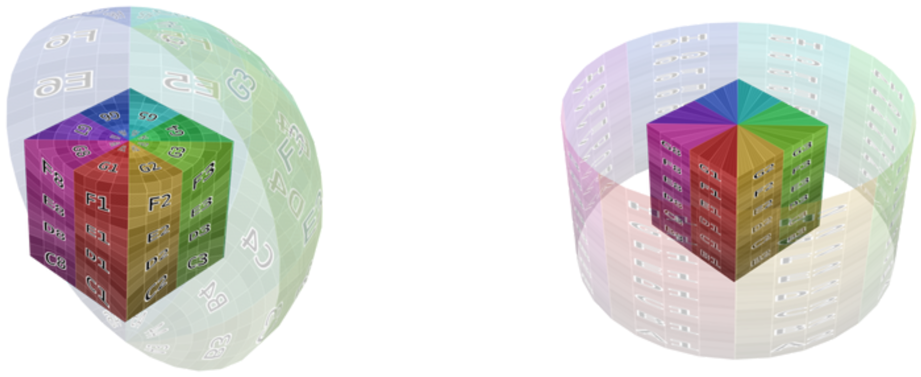
Wrap the image around a cylinder with base (0.5, 0.5, 0) and height 1, and project the *Vector* horizontally from the central axis onto this cylinder. This projection is useful for a label on a bottle, for example. However, it's not suited for the top or bottom side of objects.

Projections demonstrated using "Object" texture coordinates



Flat projection

Box projection



Sphere projection

Tube projection

Extension

How the image is extrapolated if *Vector* lies outside the regular (0, 0, 0) to (1, 1, 1) bounds:

Repeat::	Repeat the image horizontally and vertically (tiling).
Extend::	Extend the image by repeating the pixels on its edges.
Clip::	Clip to the original image size and set all the exterior pixels values to transparent black.
Mirror::	Repeatedly flip the image horizontally and vertically.

Source

Type of image (Single Image, Movie…). See Image Settings.

Frames

How many frames of the Movie-type image (video) to play. Past this point, the video will be paused (unless *Cyclic* is enabled).

If you want to play the whole video, you can click Match Movie Length in the Image Editor’s Sidebar, then copy the *Frames* from there to the node.

Start Frame

Scene frame at which the video should start playing.

Offset

Number of frames to offset the video to an earlier point in time. (Put differently: how many frames at the start of the video to skip.)

Hint

Blender plays video textures at the scene framerate, not their original framerate, meaning they’ll be faster or slower than intended if these framerates don’t match up. You can put a Driver on the Offset to work around this. Simply type the following into the field, replacing *StartFrame*, *VideoFrameRate* and *SceneFrameRate* by their respective numbers:

#(frame - StartFrame) * (VideoFrameRate - SceneFrameRate) / SceneFrameRate

Cyclic

Start over after the last frame to create a continuous loop.

Auto Refresh

Update the video texture in the 3D Viewport when moving through the timeline.

Color Space

The Color Space the image file was saved in. See Image Settings for details.

Alpha

How the image uses its Alpha Channel. See Image Settings for details.

Outputs

Color

RGB color from image. If the image has transparency, the color is premultiplied if the Alpha output is used, and unpremultiplied (straight) otherwise.

Alpha

Alpha channel from image.

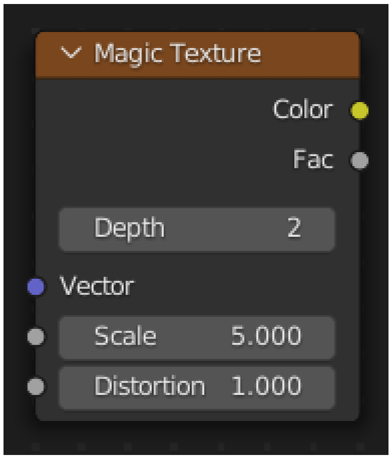
Examples

Blender - Material Preview

Image texture from GoodTextures.com.

Magic Texture Node

The Magic Texture node is used to add a psychedelic color texture. It can be used for “Thin Film Interference” if you assign a *Reflection* Texture Coordinate to the Vector input and use a relatively high *Turbulence*. The RGB components are generated independently with a sine formula.



Depth 2
Scale 5.000
Distortion 1.000

Inputs

Vector
: Texture coordinate to sample texture at; defaults to Generated texture coordinates if the socket is left unconnected.

Scale
: Scale of the texture.

Distortion
: Amount of distortion.

Properties

Depth
: Number of iterations.

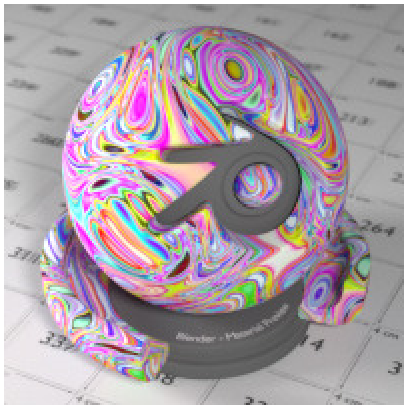
Outputs

Color

Texture color output.

Factor
Texture intensity output.

Examples



Magic texture: Depth 10, Distortion 2.0.

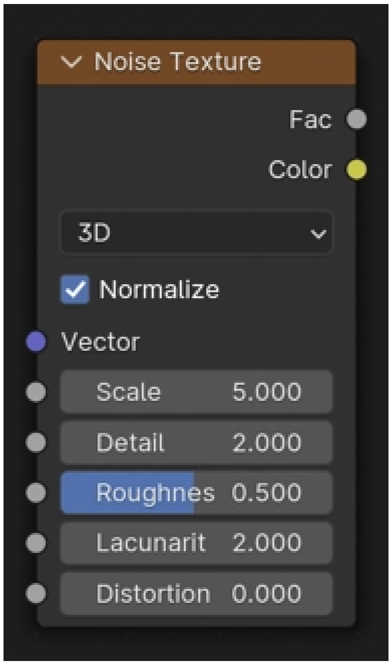
Musgrave Texture Node

The Musgrave texture node was replaced by the Noise Texture node, which includes all the same functionality.

- The Dimension input was replaced by a Roughness input, where $Roughness = Lacunarity^{-Dimension}$.
- The Detail input value must be subtracted by 1 compared to the old Musgrave Texture node.

Noise Texture Node

The *Noise Texture* node evaluates a fractal Perlin noise at the input texture coordinates. It can be used for a single Perlin noise evaluation, or for combining multiple octaves (layers) with increasingly finer detail.



3D

☑ Normalize

Scale 5.000

Detail 2.000

Roughnes 0.500

Lacunarit 2.000

Distortion 0.000

Inputs

The inputs are dynamic: they become available if needed depending on the node properties.

Vector

Texture coordinate to evaluate the noise at; defaults to *Generated* texture coordinates if the socket is left unconnected.

W

Texture coordinate to evaluate the noise at.

Scale

Scale of the base noise octave.

Detail

Number of noise octaves. This can have a fractional part, in which case a blend is performed (e.g. a Detail of 2.5 results in a 50% blend between 2 and 3 octaves).

Roughness

Blend between a smoother noise pattern, and rougher with sharper peaks.

Lacunarity

The difference between the scale of each two consecutive octaves. Larger values corresponds to larger scale for higher octaves.

Offset

An added offset to each octave, determines the level where the highest octave will appear.

Gain

An extra multiplier to tune the magnitude of octaves.

Distortion

Amount of distortion.

Properties

Dimensions

The dimensions of the space to evaluate the noise in.

1D::	Evaluate the noise in 1D space at the input *W*.
2D::	Evaluate the noise in 2D space at the input *Vector*. The Z component is ignored.
3D::	Evaluate the noise in 3D space at the input *Vector*.
4D::	Evaluate the noise in 4D space at the input *Vector* and the input *W* as the fourth dimension.

Note

Higher dimensions corresponds to higher render time, so lower dimensions should be used unless higher dimensions are necessary.

Type

Type of Noise texture, with different ways to combine octaves.

fBM::	Fractal Brownian motion, produces a homogeneous and isotropic result. Values from octaves are added together.
Multifractal::	More uneven, varying by location similar to real terrain. Values from octaves are multiplied together.
Hybrid Multifractal::	Creates peaks and valleys with different roughness values, like real mountains rise out of flat plains. Combines octaves using both addition and multiplication.
Ridged Multifractal::	Creates sharp peaks. Calculates the absolute value of the noise, creating "canyons", and then flips the surface upside down.
Hetero Terrain::	Similar to *Hybrid Multifractal* creates a heterogeneous terrain, but with the likeness of river channels.

Normalize *fBM*

If enabled, ensures that the output values stay in the range 0.0 to 1.0. If disabled, the range is at most -(*Detail* + 1) to *Detail* + 1 (smaller if *Roughness* < 1).

Outputs

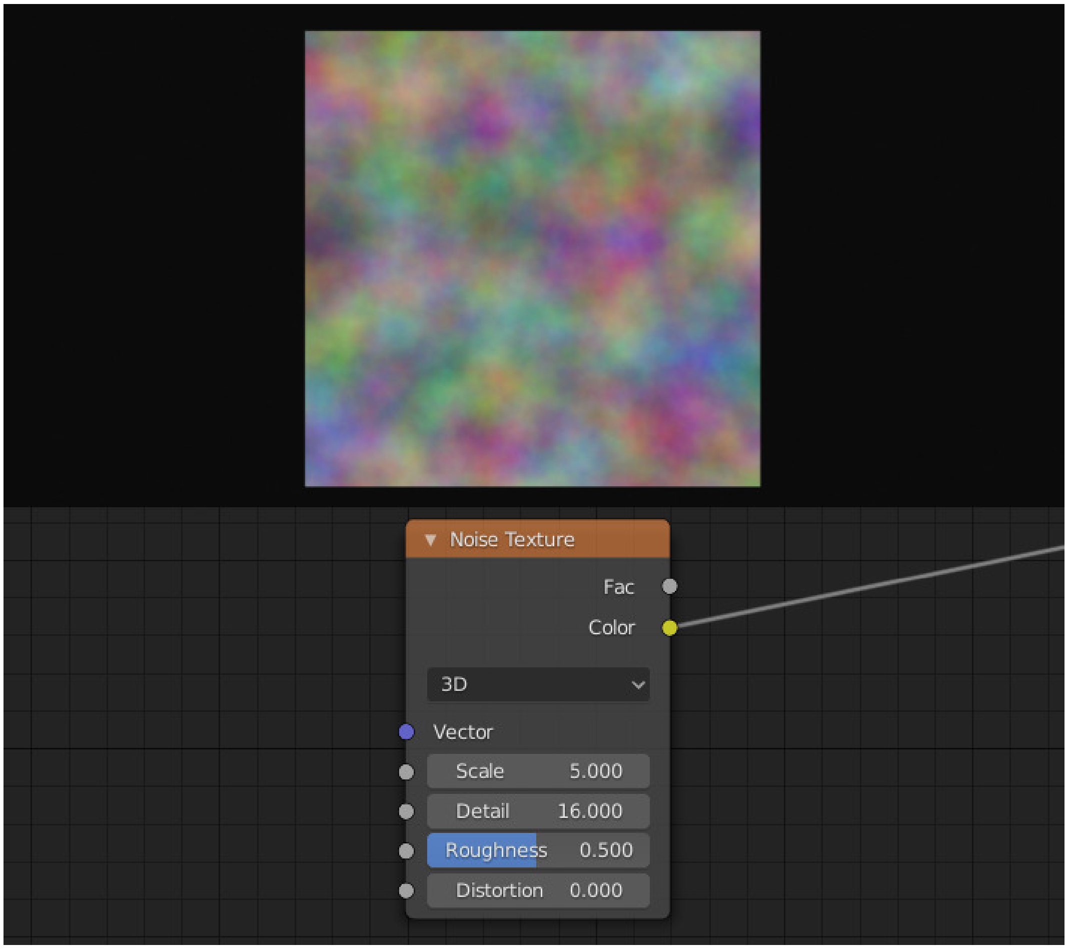
Factor

Value of fractal noise.

Color

Color with different fractal noise in each component.

Examples


Noise Texture
Fac
Color
Vector

3D

Scale 5.000

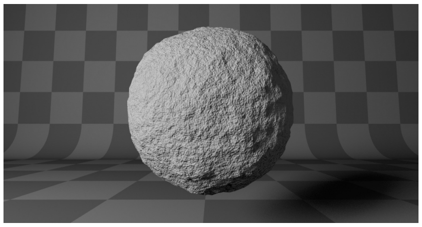
Detail 16.000

Roughness 0.500

Distortion 0.000

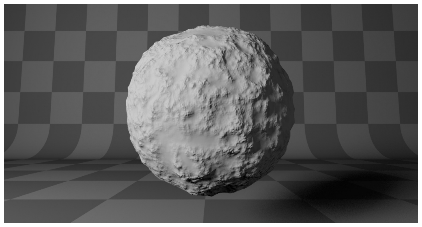
Noise Texture with high detail.

Different Noise types with the same parameters.

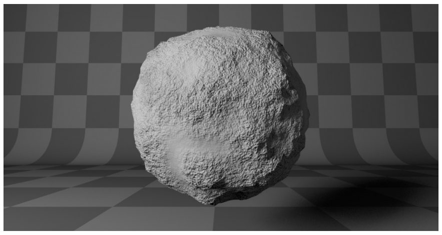


Multifractal.

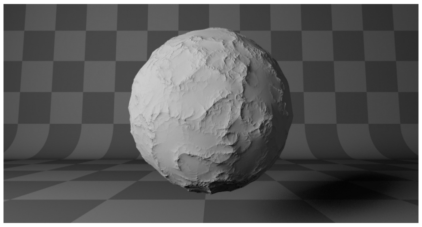
fBM (fractal Brownian Motion).



Hybrid Multifractal.



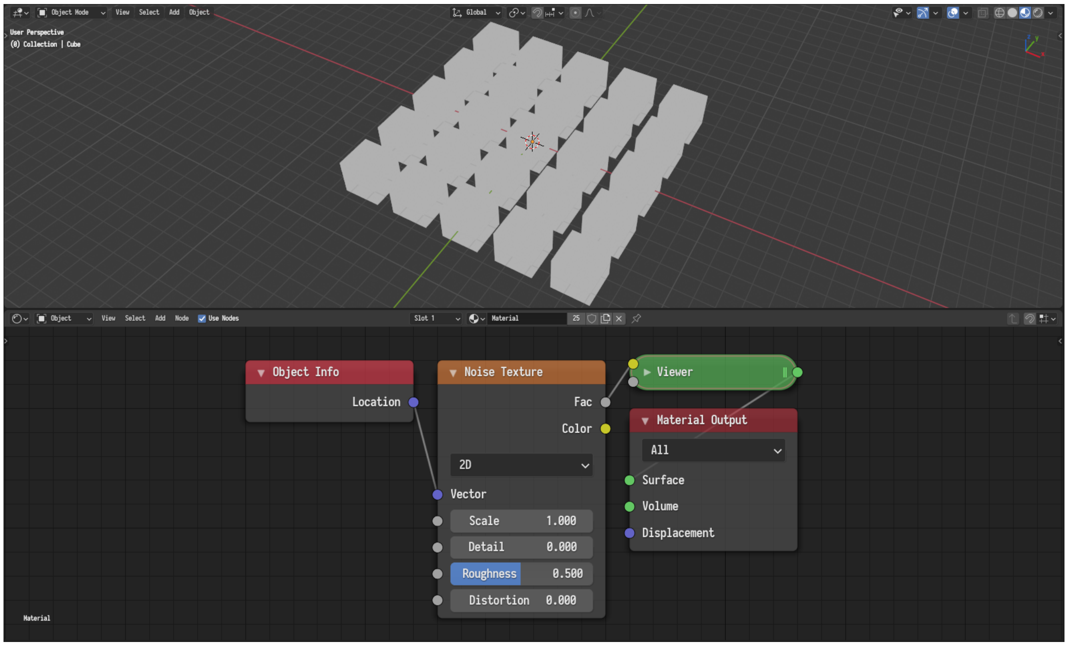
Heterogeneous Terrain.



Ridged Multifractal.

Notes

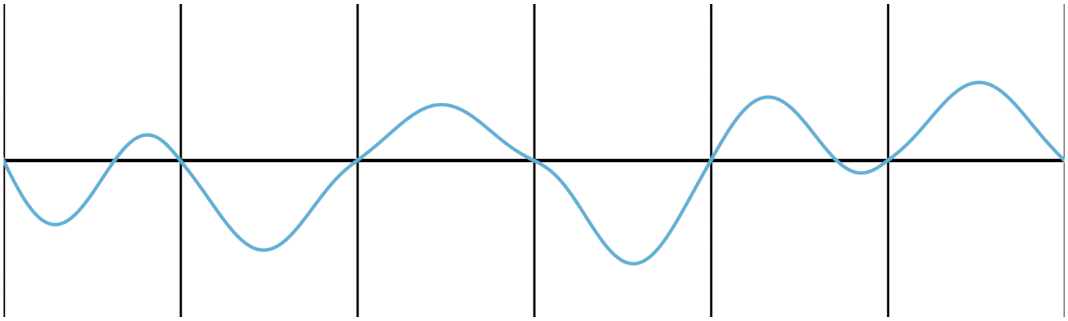
While the noise is random in nature, it follows a certain pattern that might not evaluate to random values in some configurations. For instance, consider the following configuration where a grid of objects have a material that evaluates a noise texture at their locations. One might expect the objects to have random values since they have different locations, but this is not the case.


Object Mode
View
Select
Add
Object
Global
User Perspective
(0) Collection | Cube
Object
View
Select
Add
Node
Use Nodes
Slot 1
Material
Object Info
Location
Noise Texture
Fac
Color
Vector
Viewer
Material Output
All
Surface
Volume
Displacement
Material

2D
Scale 1.000
Detail 0.000
Roughness 0.500
Distortion 0.000

An example configuration where the noise evaluates to a constant value.

It seems all objects have a value of 0.5. To understand why this happens, let us look at the following plot of a 1D noise texture.



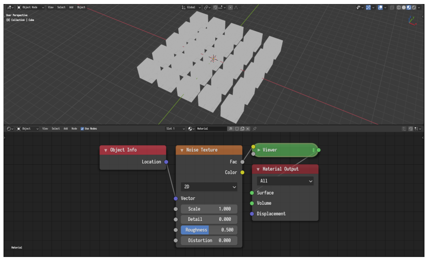
A plot of a 1D noise with zero details and zero distortion.

The horizontal line denotes a value of 0.5 and the vertical lines denotes whole numbers assuming a noise scale of 1. As can be seen, the noise always intersects the 0.5 line at whole numbers. Since the aforementioned objects were distributed on a grid and have whole number locations, they all evaluate to 0.5. Which explains the issue at hand.

Generally, any discrete evaluation of noise at integer multiples of the reciprocal of the noise scale will always evaluate to 0.5. It also follows that evaluations closer to that will have values close to 0.5. In such cases, it is almost always preferred to use the White Noise Texture.

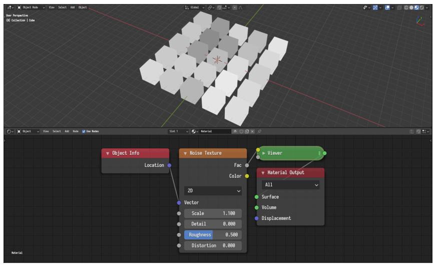
Regardless, one can mitigate this issue in a number of ways:

- Adjust the scale of the noise to avoid aligning the noise with the evaluation domain.
- Add an arbitrary offset to the texture coordinates to break the alignment with the evaluation domain.
- Evaluate the noise at a higher dimension and adjust the extra dimension until a satisfactory result is achieved.


Add
Node
Use Nodes
Slot 1
Material
Object Info
Location
Noise Texture
Fac
Color
Vector
Viewer
Material Output
Surface
Volume
Displacement

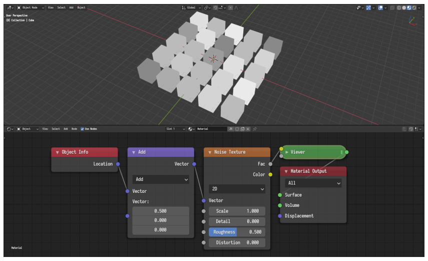
2D
Scale 1.000
Detail 0.000
Roughness 0.500
Distortion 0.000
All

Constant value issue.


Add
Node
Use Nodes
Slot 1
Material
Object Info
Location
Noise Texture
Fac
Color
Vector
Viewer
Material Output
Surface
Volume
Displacement

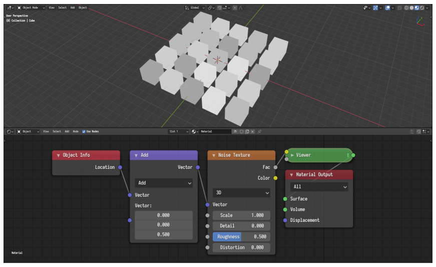
2D
Scale 1.100
Detail 0.000
Roughness 0.500
Distortion 0.000
All

Mitigating the issue by adjusting the scale.


Object
View
Select
Add
Node
Use Nodes
Slot 1
Material
Object Info
Location
Add
Vector
Noise Texture
Fac
Color
Vector
Viewer
Material Output
Surface
Volume
Displacement

Add
Vector:
0.500
0.000
0.000
2D
Scale 1.000
Detail 0.000
Roughness 0.500
Distortion 0.000
All

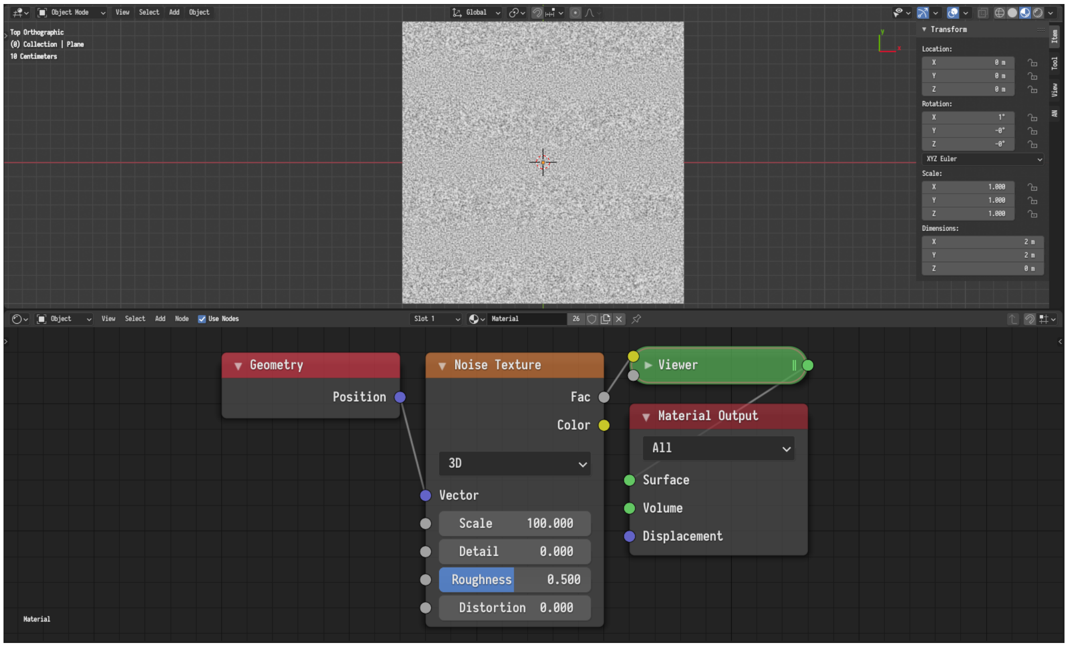
Mitigating the issue by adding an arbitrary offset.


Object
View
Select
Add
Node
Use Nodes
Slot 1
Material
Object Info
Location
Add
Vector
Noise Texture
Fac
Color
Vector
Viewer
Material Output
Surface
Volume
Displacement

Add
Vector:
0.000
0.000
0.500
3D
Scale 1.000
Detail 0.000
Roughness 0.500
Distortion 0.000
All

Mitigating the issue by evaluating at a higher dimension.

Similarly, in other configurations, one might experience some banding patterns in the noise, where there are bands of high contrast areas followed by banding of low contrast areas. For instance, planar surfaces that are slightly tilted along one of the axis will have such a banding pattern.


Object Mode
View
Select
Add
Object
Global
Top Orthographic
(0) Collection | Plane
10 Centimeters
Item
Tool
View
Object
View
Select
Add
Node
☑ Use Nodes
Slot 1
Material
Geometry
Noise Texture
Viewer
Material Output
Fac
Color
Vector
Surface
Volume
Displacement
Material

Transform

Location:
X 0 m
Y 0 m
Z 0 m

Rotation:
X 1°
Y -0°
Z -0°
XYZ Euler

Scale:
X 1.000
Y 1.000
Z 1.000

Dimensions:
X 2 m
Y 2 m
Z 0 m

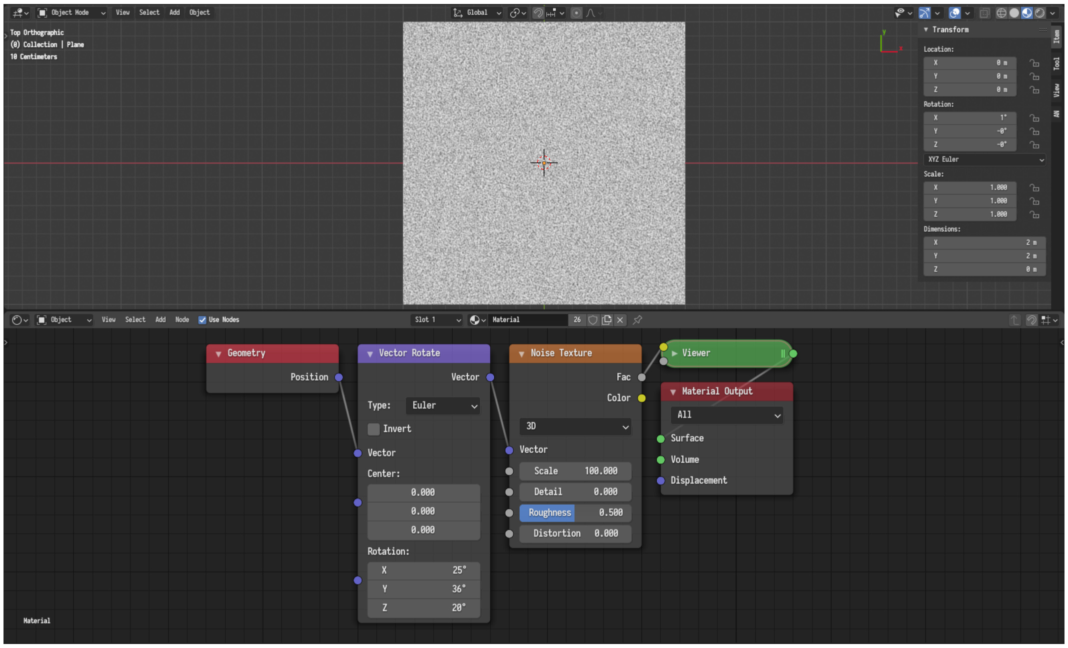
Geometry: Position → Noise Texture: Vector

Noise Texture: 3D
Scale 100.000
Detail 0.000
Roughness 0.500
Distortion 0.000

Material Output: All

An example configuration where the noise have a banding pattern.

This happens because the slight tilt along one of the axis causes values along the perpendicular axis to change very slowly making the grid structure of the noise more apparent. The easiest way to mitigate this issue to rotate the coordinates by an arbitrary amount.


Object Mode
View
Select
Add
Object
Global
Top Orthographic
(0) Collection | Plane
10 Centimeters
Transform
Item
Tool
View
Object
Node
Use Nodes
Slot 1
Material
Geometry
Position
Vector Rotate
Noise Texture
Fac
Color
Viewer
Material Output
Surface
Volume
Displacement
Material

Location: X 0 m, Y 0 m, Z 0 m

Rotation: X 1°, Y -0°, Z -0°

XYZ Euler

Scale: X 1.000, Y 1.000, Z 1.000

Dimensions: X 2 m, Y 2 m, Z 0 m

Vector Rotate: Type: Euler

☐ Invert

Vector

Center: 0.000, 0.000, 0.000

Rotation: X 25°, Y 36°, Z 20°

Noise Texture: 3D

Vector

Scale 100.000

Detail 0.000

Roughness 0.500

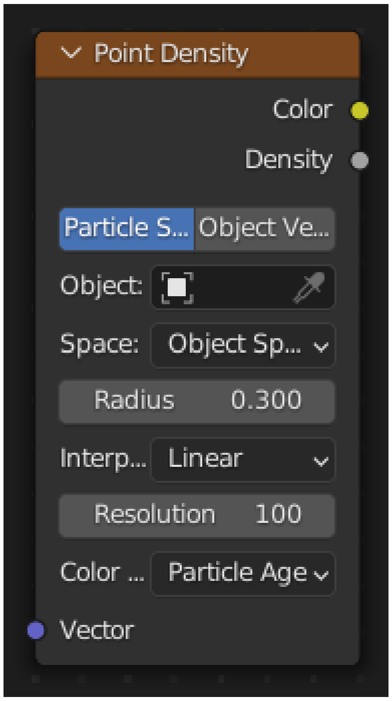
Distortion 0.000

Material Output: All

Mitigating the issue by rotating the coordinates by an arbitrary amount.

Point Density Node

The *Point Density* node is available in volume shaders, to render volumetric points for each particle or vertex of another object.



Object:

Space: Object Sp...

Radius 0.300

Interp... Linear

Resolution 100

Color ... Particle Age

Inputs

Vector

> Texture coordinate to sample texture at; defaults to global position (*Position* output of *Geometry* node) if the socket is left unconnected.

Properties

Point Data

> Where to get points from.
>
> Particle System
>
> > Use each particle position from the specified particle system.
>
> Object Vertices

Use each vertex position from the specified object.

Object

Which object's vertices or particle system will be used.

Particle System

Particle positions from this system will be used.

Space

The coordinate system for mapping points.

World Space::	Map each point exactly where the source particle or vertex is.
Object Space::	Fit the points from the source particles/vertices inside the bounding box of the object with the point density texture.

Radius

Size of the points.

Interpolation

Texel filtering type.

Closest::	No interpolation, use nearest texel. Produces blocky looking points.
Linear::	Interpolate linearly between texels, producing soft, round points.
Cubic::	Use cubic falloff, producing very soft points. Useful when points are very densely packed.

Resolution

The dimensions of the texture holding the point data.

Color Source

Which attribute of the particle system or mesh is used to color the output.

Particle Color Sources

Particle Age::	Lifetime mapped as (0.0 - 1.0) intensity.
Particle Speed::	Particle speed (absolute magnitude of velocity) mapped as (0.0 - 1.0) intensity.
Particle Velocity::	XYZ velocity mapped to RGB colors.

Vertex Color Sources

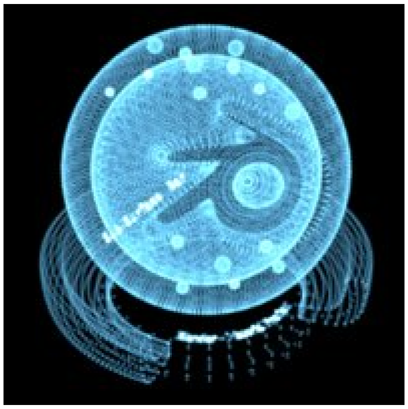
Vertex Color::	Use a Color Attribute for coloring the point density texture. **Note** Color Attributes are defined per face corner. A single vertex can have as many different colors as faces it is part of. The actual color of the point density texture is averaged from all vertex corners.
Vertex Weight::	Use weights from a vertex group as intensity values.
Vertex Normals::	Use object-space vertex normals as RGB values.

Outputs

Color
: Texture color output.

Density
: Density of volume.

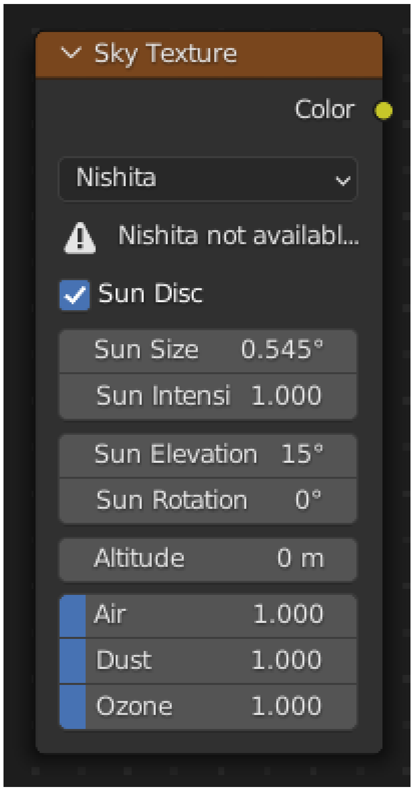
Examples



Domain object with Point Density texture using vertices from ball as points.

Sky Texture Node

The *Sky Texture* node generates a procedural sky. It's typically used in combination with the World Output Node.



Sky Texture

Color

Nishita

Nishita not availabl...

☑ Sun Disc

Sun Size 0.545°

Sun Intensi 1.000

Sun Elevation 15°

Sun Rotation 0°

Altitude 0 m

Air 1.000

Dust 1.000

Ozone 1.000

Inputs

Vector
: Texture coordinate to sample texture at; defaults to Generated texture coordinates if the socket is left unconnected.

Properties

Sky Type
: Sky model to use.

  Preetham
  : Based on the 1999 paper by Preetham et al.

Hosek/Wilkie

Based on the 2012 paper by Hosek and Wilkie.

Nishita

Improved version of the 1993 model by Nishita et al.

Note that this sky type is quite bright and makes the image look overexposed with the default scene settings. You can reduce the Exposure setting in *Properties* ▸ *Render* ▸ *Film* to fix this.

Sun Direction

Sun direction vector.

Turbidity

Atmospheric turbidity.

- 2: Arctic like
- 3: clear sky
- 6: warm/moist day
- 10: hazy day

Ground Albedo

Amount of light reflected from the planet surface back into the atmosphere.

Sun Disc *Cycles Only*

Enable/Disable sun disc lighting.

Sun Size

Angular diameter of the sun disc (in degrees).

Sun Intensity

Multiplier for sun disc lighting.

Sun Elevation

Rotation of the sun from the horizon (in degrees).

Sun Rotation

Rotation of the sun around the zenith (in degrees).

Altitude

The distance from sea level to the location of the camera. For example, if the camera is placed on a beach then a value of 0 should be used. However, if the camera is in the cockpit of a flying airplane then a value of 10 km will be more suitable. Note, this is limited to 60 km because the mathematical model only accounts for the first two layers of the earth's atmosphere (which ends around 60 km).

Air

Density of air molecules.

- 0 no air
- 1 clear day atmosphere
- 2 highly polluted day

Dust

Density of dust and water droplets.

- 0 no dust
- 1 clear day atmosphere
- 5 city like atmosphere
- 10 hazy day

Ozone

Density of ozone molecules; useful to make the sky appear bluer.

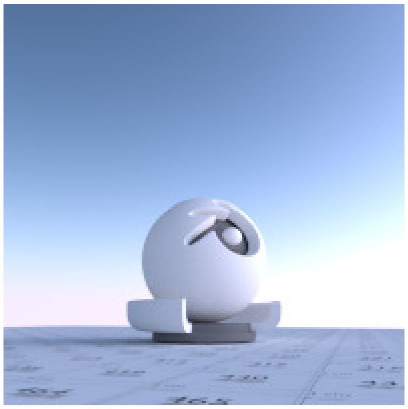
- 0 no ozone
- 1 clear day atmosphere
- 2 city like atmosphere

Outputs

Color

Texture color output.

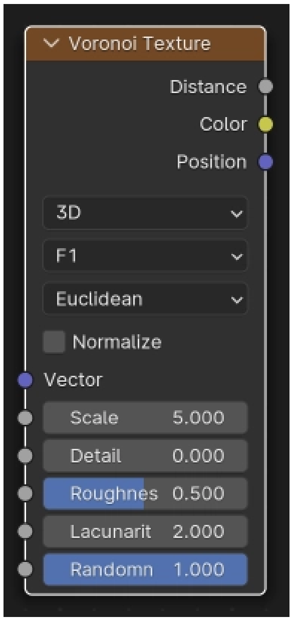
Examples



Example of Sky Texture.

Voronoi Texture Node

The *Voronoi Texture* node evaluates a Worley Noise at the input texture coordinates.



3D

F1

Euclidean

☐ Normalize

Scale 5.000

Detail 0.000

Roughnes 0.500

Lacunarit 2.000

Randomn 1.000

Inputs

The inputs are dynamic, they become available if needed depending on the node properties.

Vector
: Texture coordinate to evaluate the noise at; defaults to *Generated* texture coordinates if the socket is left unconnected.

W
: Texture coordinate to evaluate the noise at.

Scale
: Scale of the noise.

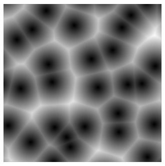
Detail
: Number of noise octaves. The fractional part of the input is multiplied by the magnitude of the highest octave. Higher number of octaves corresponds to a higher evaluation time.

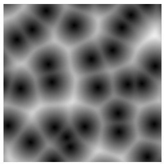
Roughness
: Blend between a smoother noise pattern, and rougher with sharper peaks.

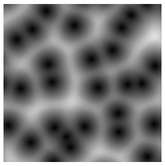
Lacunarity
: The difference between the scale of each two consecutive octaves. Larger values corresponds to larger scale for higher octaves.

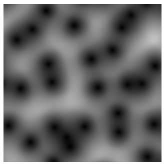
Smoothness

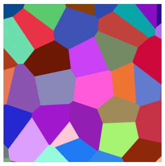
The smoothness of the noise.

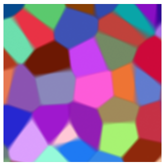

Smoothness: 0.0.

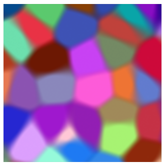

Smoothness: 0.25.

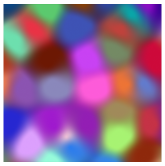

Smoothness: 0.5.


Smoothness: 1.0.


Smoothness: 0.0.

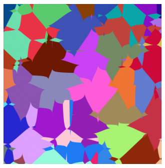

Smoothness: 0.25.

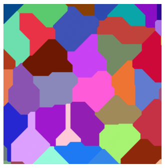

Smoothness: 0.5.

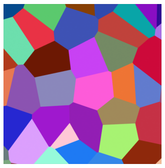

Smoothness: 1.0.

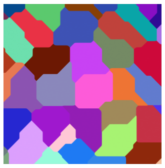
Exponent

Exponent of the Minkowski distance metric.


Exponent: 0.5.

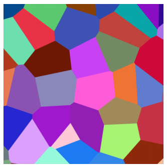

Exponent: 1.0.

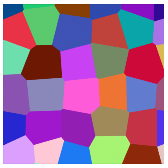

Exponent: 2.0.

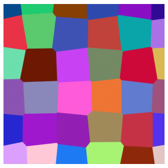

Exponent: 32.0.

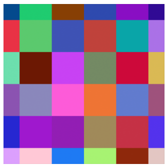
Randomness

The randomness of the noise.


Randomness: 1.0.


Randomness: 0.5.


Randomness: 0.25.


Randomness: 0.0.

Properties

Dimensions

The dimensions of the space to evaluate the noise in.

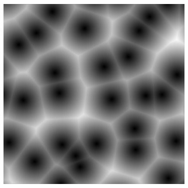
1D::	Evaluate the noise in 1D space at the input W.
2D::	Evaluate the noise in 2D space at the input Vector. The Z component is ignored.
3D::	Evaluate the noise in 3D space at the input Vector.
4D::	Evaluate the noise in 4D space at the input Vector and the input W as the fourth dimension.

Higher dimensions corresponds to higher render time, so lower dimensions should be used unless higher dimensions are necessary.

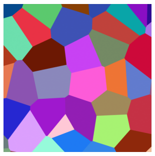
Feature

The Voronoi feature that the node will compute.

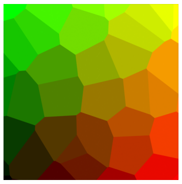
F1:: The distance to the closest feature point as well as its position and color.



Distance.

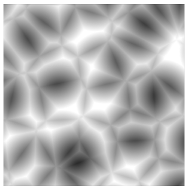


Color.

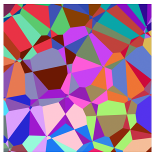


Position.

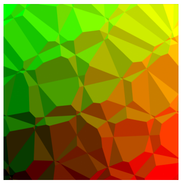
F2:: The distance to the second closest feature point as well as its position and color.



Distance.

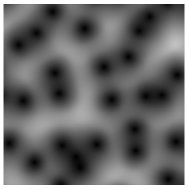


Color.

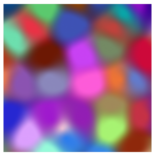


Position.

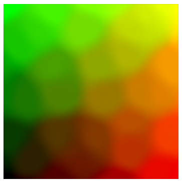
Smooth F1:: A smooth version of F1.



Distance.

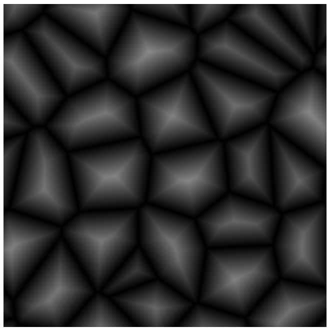


Color.

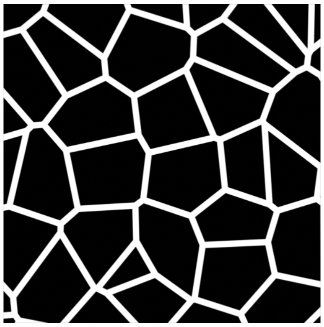


Position.

Distance to Edge:: The distance to the edges of the Voronoi cells.

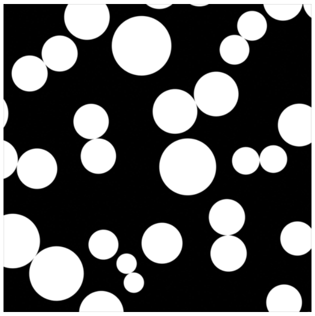


Distance.

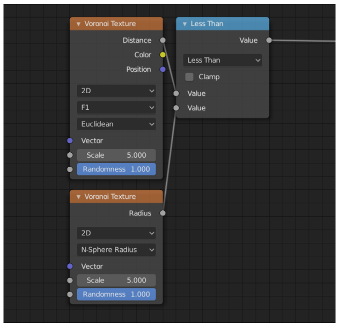


Distance smaller than 0.05.

N-Sphere Radius:: The radius of the n-sphere inscribed in the Voronoi cells. In other words, it is half the distance between the closest feature point and the feature point closest to it.



The n-sphere radius can be used to create tightly packed n-spheres.



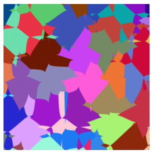
Voronoi Texture
Distance
Color
Position
2D
F1
Euclidean
Vector
Scale 5.000
Randomness 1.000
Less Than
Value
Less Than
Clamp
Value
Value
Voronoi Texture
Radius
2D
N-Sphere Radius
Vector
Scale 5.000
Randomness 1.000

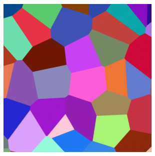
Node tree for the shader to the left.

Distance Metric

The distance metric used to compute the texture.

Euclidean::	Use the Euclidean distance metric.
Manhattan::	Use the Manhattan distance metric.
Chebychev::	Use the Chebychev distance metric.
Minkowski::	Use the Minkowski distance metric. The Minkowski distance is a generalization of the aforementioned metrics with an *Exponent* as a parameter. Minkowski with an exponent of one is equivalent to the *Manhattan* distance metric. Minkowski with an exponent of two is equivalent to the *Euclidean* distance metric. Minkowski with an infinite exponent is equivalent to the *Chebychev* distance metric.





Minkowski Exponent: 0.5 (Minkowski 1/2).	Minkowski Exponent: 1.0 (Manhattan).	Minkowski Exponent: 2.0 (Euclidean).	Minkowski Exponent: 32.0 (approximation of Chebychev).

Normalize

If enabled, ensures that the output values stay in the range 0.0 to 1.0. In rare cases, the output value may be outside that range when *Feature* is *F2*.

Outputs

Distance

Distance.

Color

Cell color. The color is arbitrary.

Position

Position of feature point.

W

Position of feature point.

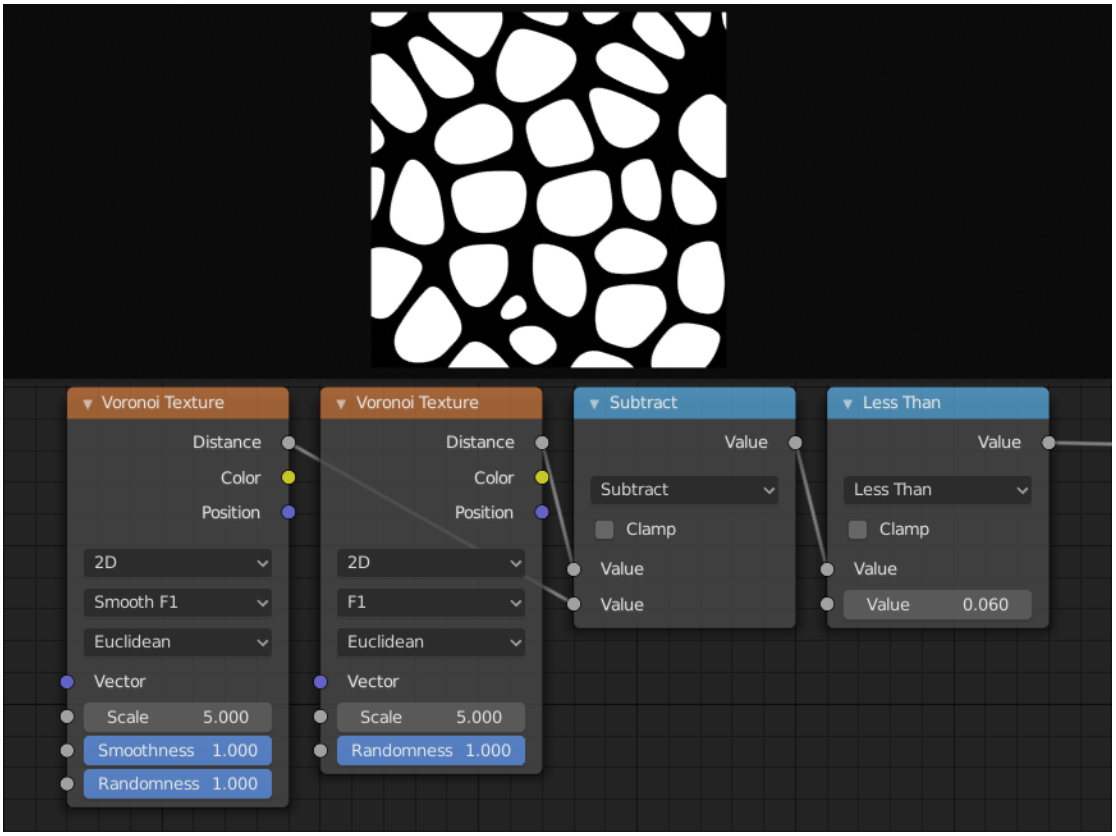
Radius

N-Sphere radius.

Notes

In some configurations of the node, especially for low values of *Randomness*, rendering artifacts may occur. This happens due to the same reasons described in the Notes section in the White Noise Texture page and can be fixed in a similar manner as described there.

Examples


Voronoi Texture
Distance
Color
Position
Vector
Voronoi Texture
Distance
Color
Position
Vector
Subtract
Value
Value
Value
Less Than
Value
Value

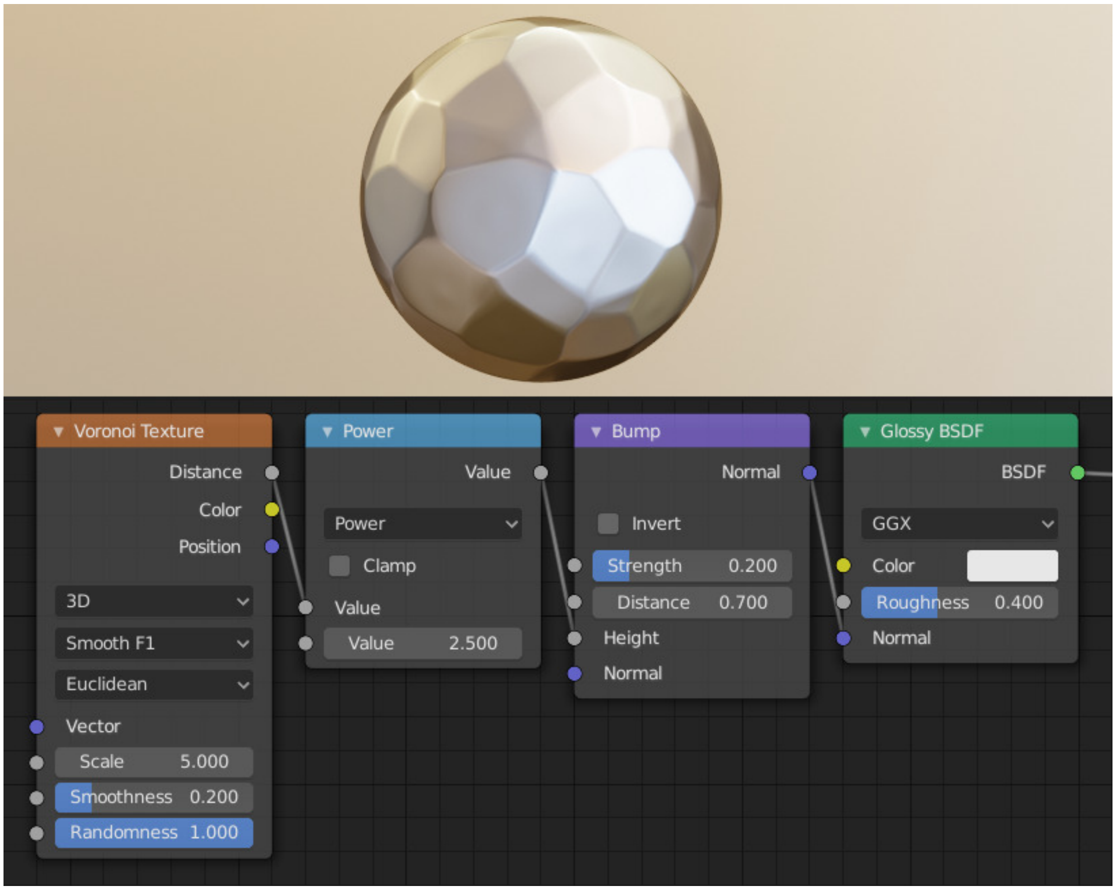
Voronoi Texture: 2D, Smooth F1, Euclidean, Scale 5.000, Smoothness 1.000, Randomness 1.000

Voronoi Texture: 2D, F1, Euclidean, Scale 5.000, Randomness 1.000

Subtract: Subtract, ☐ Clamp

Less Than: Less Than, ☐ Clamp, Value 0.060

The difference between *F1* and *Smooth F1* can be used to create beveled Voronoi cells.


Voronoi Texture
Distance
Color
Position
Power
Value
Bump
Normal
Height
Glossy BSDF
BSDF
Color
Normal
Vector

Voronoi Texture: 3D, Smooth F1, Euclidean, Scale 5.000, Smoothness 0.200, Randomness 1.000

Power: Power, ☐ Clamp, Value 2.500

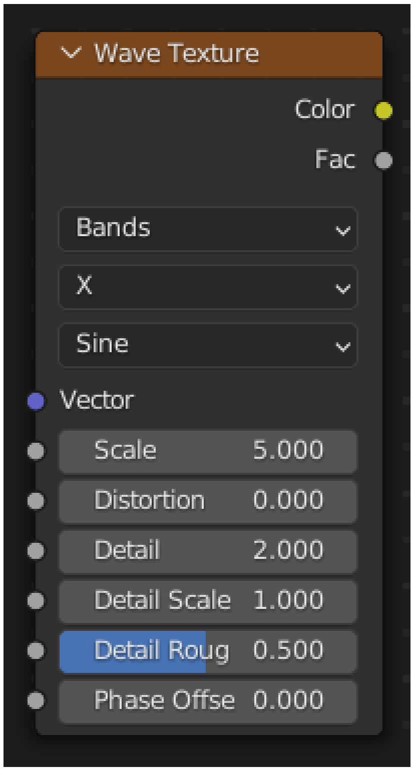
Bump: ☐ Invert, Strength 0.200, Distance 0.700

Glossy BSDF: GGX, Roughness 0.400

Creating a hammered metal shader using the *Voronoi Texture* node.

Wave Texture Node

The *Wave Texture* node adds procedural bands or rings with noise distortion.


Wave Texture

Color

Fac

Bands

X

Sine

Vector

Scale 5.000

Distortion 0.000

Detail 2.000

Detail Scale 1.000

Detail Roug 0.500

Phase Offse 0.000

Inputs

Vector
: Texture coordinate to sample texture at; defaults to Generated texture coordinates if the socket is left unconnected.

Scale
: Overall texture scale.

Distortion
: Amount of distortion of the wave.

Hint

> In general, textures can be distorted by mixing their texture coordinates with another texture. The distortion built into the *Wave Texture Node* uses the *Color* output of the Noise Texture Node.
>
> To replicate this, center its value range around zero, multiply it by a factor proportional to *Distortion/Scale* and add the result onto the texture coordinates. *Detail*, *Detail Scale*, and *Roughness* of the *Wave Texture Node* correspond to the inputs on the Noise Texture Node.

Detail
: Amount of distortion noise detail.

Detail Scale
: Scale of distortion noise.

Roughness
: Blend between a smoother noise pattern, and rougher with sharper peaks.

Phase Offset
: Position of the wave along the *Bands Direction*. This can be used as an input for more control over the distortion.

Properties

Type
: *Bands* or *Rings* shaped waves.

Bands/Rings Direction
: The axis the bands or rings propagate from i.e. which axis they are perpendicular to. When using *Bands* a *Diagonal* axis is an option and when using *Rings* the rings can propagate outwards from a single point by using *Spherical* direction.

Wave Profile
: Controls the look of the wave type.

Saw::	Uses a sawtooth profile.
Sine::	Uses the standard sine profile.

Outputs

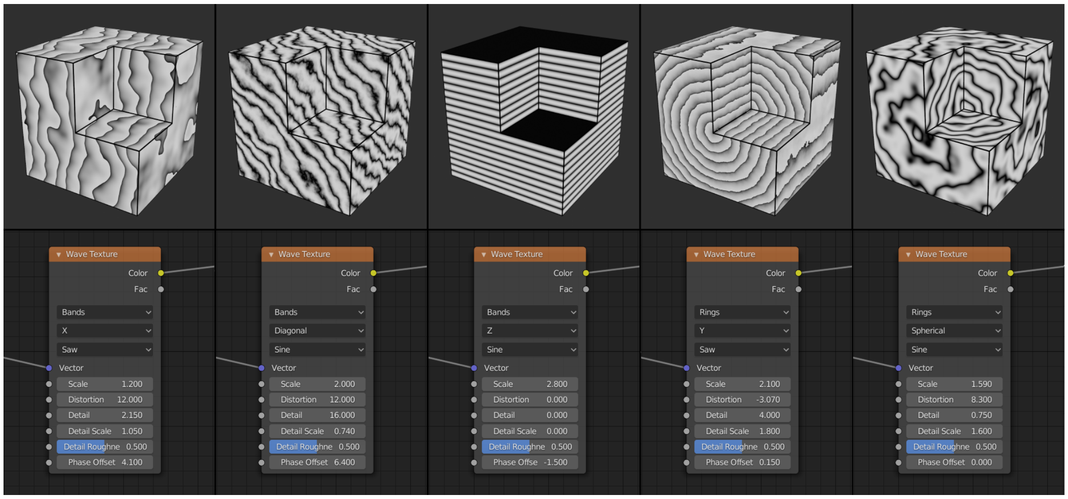
Color

Texture color output.

Factor

Texture intensity output.

Examples



Bands
X
Saw
Scale 1.200
Distortion 12.000
Detail 2.150
Detail Scale 1.050
Detail Roughne 0.500
Phase Offset 4.100

Bands
Diagonal
Sine
Scale 2.000
Distortion 12.000
Detail 16.000
Detail Scale 0.740
Detail Roughne 0.500
Phase Offset 6.400

Bands
Z
Sine
Scale 2.800
Distortion 0.000
Detail 0.000
Detail Scale 0.000
Detail Roughne 0.500
Phase Offse -1.500

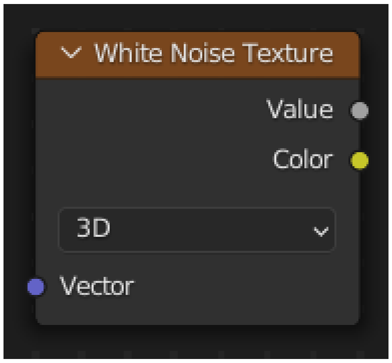
Rings
Y
Saw
Scale 2.100
Distortion -3.070
Detail 4.000
Detail Scale 1.800
Detail Roughne 0.500
Phase Offset 0.150

Rings
Spherical
Sine
Scale 1.590
Distortion 8.300
Detail 0.750
Detail Scale 1.600
Detail Roughne 0.500
Phase Offset 0.000

Wave Texture.

White Noise Texture Node

The *White Noise Texture* node returns a random number based on an input Seed. The seed can be a number, a 2D vector, a 3D vector, or a 4D vector; depending on the *Dimensions* property. The output number ranges between zero and one.



Inputs

The inputs are dynamic, they become available if needed depending on the node properties.

Vector
: Vector used as seed in 2D, 3D, and 4D dimensions.

W
: Value used as seed in 1D and 4D dimensions.

Properties

Dimensions
: The dimensions of the space to evaluate the noise in.

1D::	The *W* input is used as seed.
2D::	

The X and Y components of the *Vector* input are used as seed.

3D:: The *Vector* input is used as seed.

4D:: Both the *Vector* input and the *W* input are used as seed.

Outputs

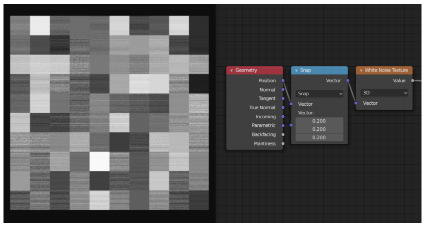
Value
: Output random value.

Color
: Output random color.

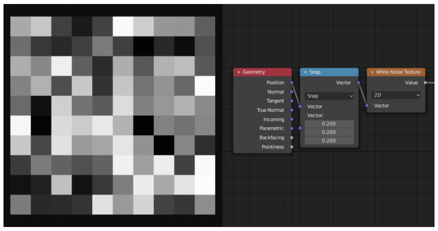
Notes

The slightest difference in seed values would result in completely different outputs. Consequently, bad precision may have significant impact on the output. Usually, we can mitigate this issue by:

- Eliminating the problematic seed value. If the problematic seed value is constant, it should be eliminated by choosing a lower dimension or multiplying it by zero.
- Adding an arbitrary value to the seed. The issue might only happen at certain boundaries, like unit boundaries, so simply adding an arbitrary value might solve the issue.
- Taking the absolute value of the seed. In computing, zero may be positive or negative, so taking the absolute values unifies the zero into a single value.


Geometry
Position
Normal
Tangent
True Normal
Incoming
Parametric
Backfacing
Pointiness
Snap
Vector
Vector
White Noise Texture
Value
Vector

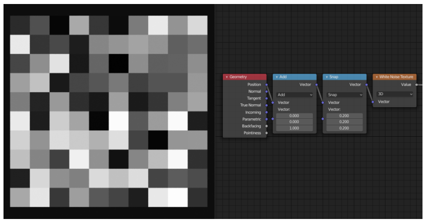
Snap
Vector: 0.200 0.200 0.200
3D


Geometry
Position
Normal
Tangent
True Normal
Incoming
Parametric
Backfacing
Pointiness
Snap
Vector
Vector
White Noise Texture
Value
Vector

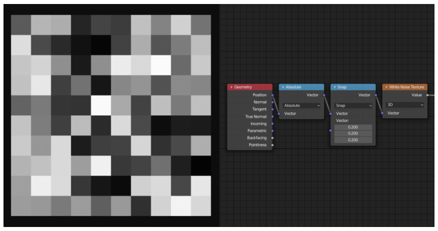
Snap
Vector: 0.200 0.200 0.200
2D

Precision issue due to signed zeros on the Z axis.

Mitigating the issue by eliminating the Z axis.

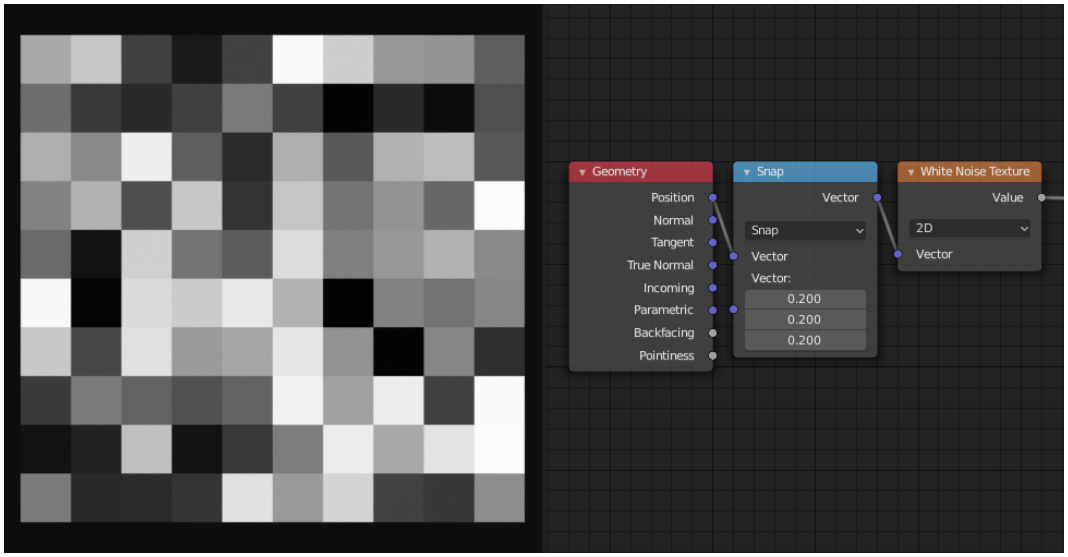


Mitigating the issue by adding an arbitrary value.



Mitigating the issue by taking the absolute value.

Examples

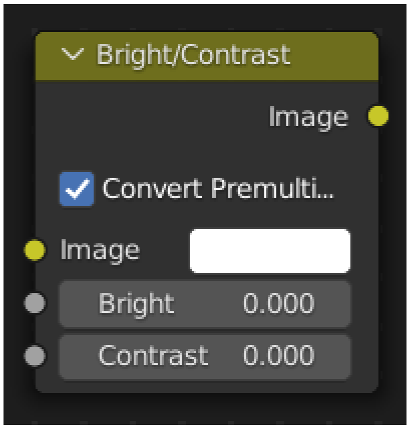

Geometry
Position
Normal
Tangent
True Normal
Incoming
Parametric
Backfacing
Pointiness
Snap
Vector
Snap
Vector
Vector:
0.200
0.200
0.200
White Noise Texture
Value
2D
Vector

Generating cell noise using the *Snap* vector operation and the *White Noise* node.

Color

- Brightness/Contrast Node
- Gamma Node
- Hue/Saturation/Value Node
- Invert Color Node
- Light Falloff Node
- Mix Color Node
- RGB Curves Node

Brightness/Contrast Node



☑ Convert Premulti...

Bright 0.000

Contrast 0.000

Inputs

Image
: Standard color input.

Brightness
: An additive-type factor by which to increase the overall brightness of the image. Use a negative number to darken an image.

Contrast
: A scaling type factor by which to make brighter pixels brighter, but keeping the darker pixels dark. Higher values make details stand out. Use a negative number to decrease the overall contrast in the image.

Properties

Convert Premultiplied
: By default, it is supposed to work in *premultiplied* alpha. If the *Convert Premul* checkbox is not enabled, it is supposed to work in *straight* alpha.

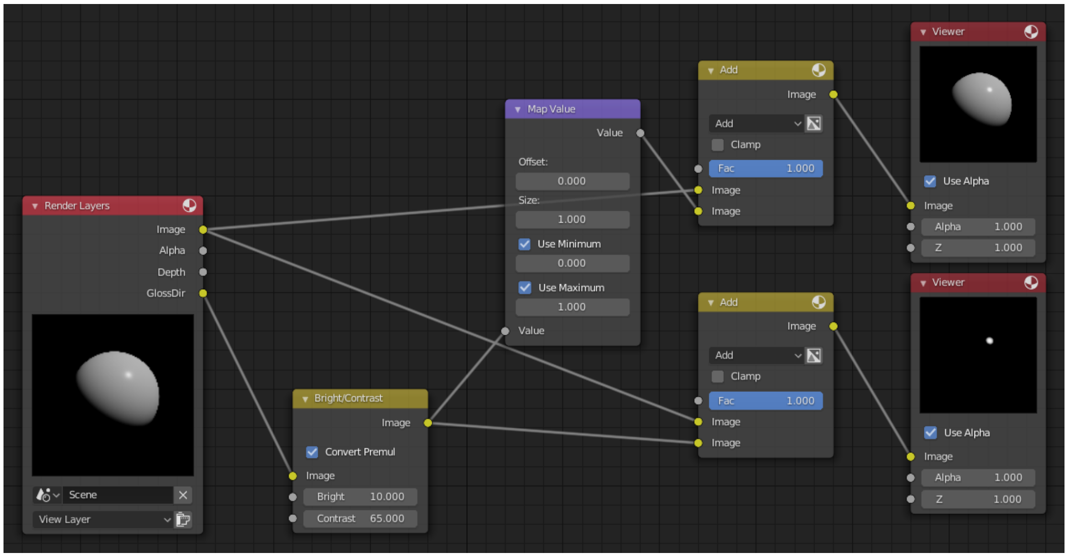
See [Alpha Channel](#).

Outputs

Image
: Standard color output.

Notes

It is possible that this node will put out a value set that has values beyond the normal range, i.e. values greater than one and less than zero. If you will be using the output to mix with other images in the normal range, you should clamp the values using the Map Value node (with the Min and Max enabled), or put through a Color Ramp node (with all normal defaults).


Render Layers
Image
Alpha
Depth
GlossDir
Scene
View Layer
Map Value
Value
Value
Bright/Contrast
Image
Image
Add
Image
Add
Image
Image
Viewer
Image
Add
Image
Add
Image
Image
Viewer
Image

Map Value: Offset: 0.000; Size: 1.000; ☑ Use Minimum 0.000; ☑ Use Maximum 1.000

Bright/Contrast: ☑ Convert Premul; Bright 10.000; Contrast 65.000

Add: Add; ☐ Clamp; Fac 1.000

Viewer: ☑ Use Alpha; Alpha 1.000; Z 1.000

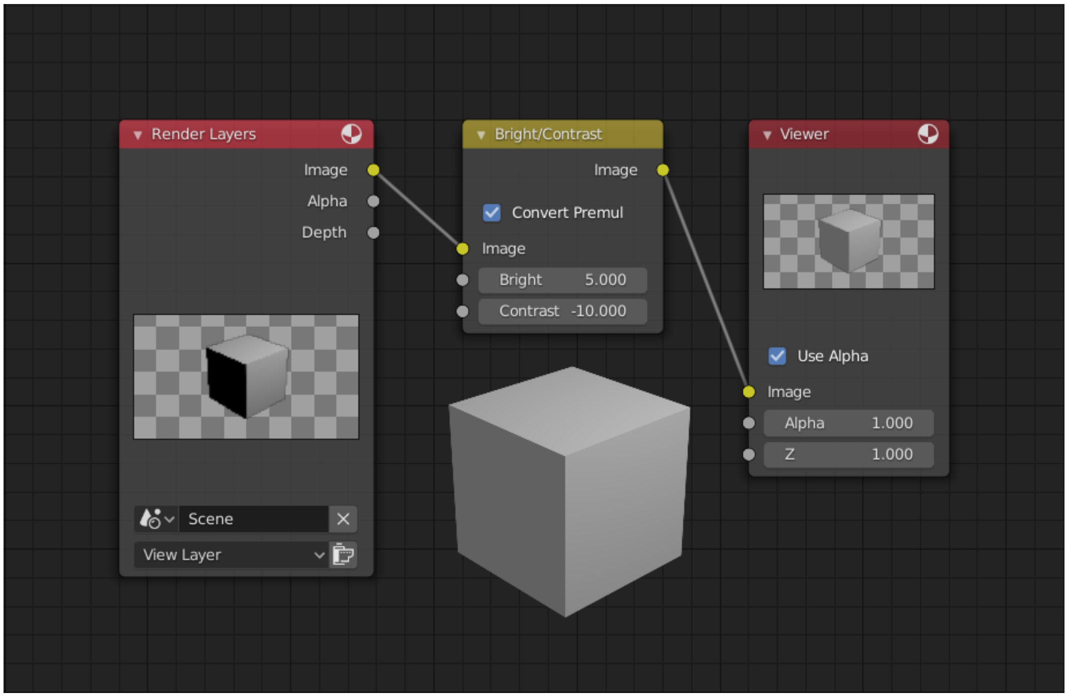
Add: Add; ☐ Clamp; Fac 1.000

Viewer: ☑ Use Alpha; Alpha 1.000; Z 1.000

Clamp the values to normal range.

Either of these nodes will scale the values back to normal range. In the example image, we want to intensify the specular pass. The bottom thread shows what happens if we do not clamp the values; the specular pass has a value much less than one in the dark areas; when added to the medium gray, it makes black. Passing the brightened image through either the Map Value or the Color Ramp node produces the desired effect.

Example


Render Layers
Image
Alpha
Depth
Scene
View Layer
Bright/Contrast
Image
Image
Viewer
Image

☑ Convert Premul

Bright 5.000

Contrast -10.000

☑ Use Alpha

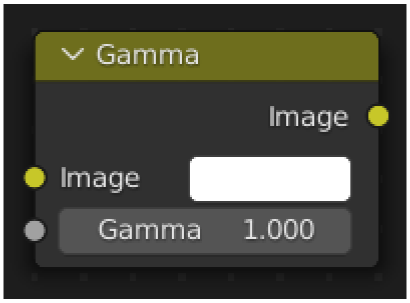
Alpha 1.000

Z 1.000

A basic example.

Gamma Node

Use this node to apply a gamma correction.


Gamma
Image
Image
Gamma 1.000

Inputs

Image
: Standard color input.

Gamma
: An exponential brightness factor.

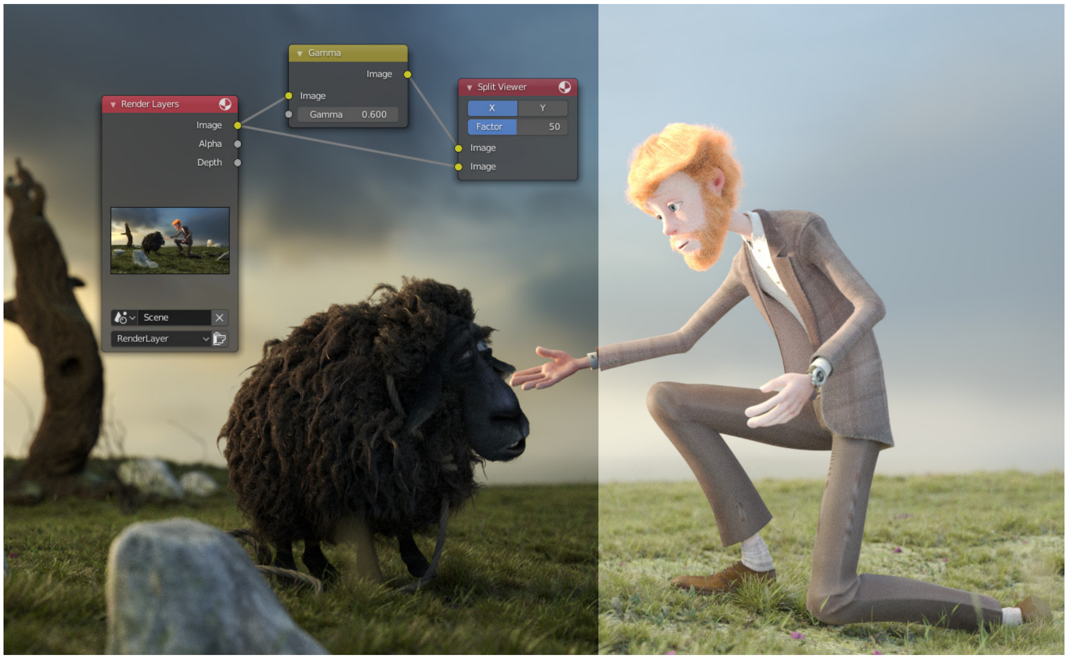
Properties

This node has no properties.

Outputs

Image
: Standard color output.

Examples


Render Layers
Image
Alpha
Depth
Scene
RenderLayer
Gamma
Image
Image
Split Viewer
X
Y
Image
Image

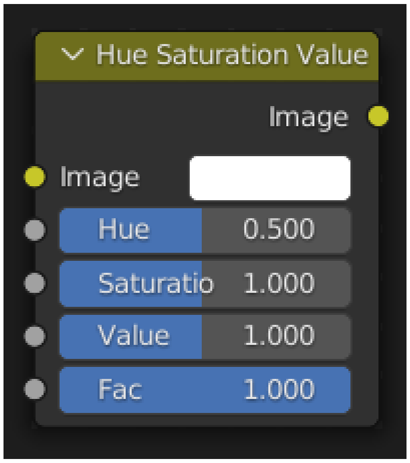
Gamma 0.600

Factor 50

Example of a Gamma node.

Hue/Saturation/Value Node

The *Hue/Saturation/Value Node* applies a color transformation in the HSV Color Model.


Hue Saturation Value
Image
Image
Hue 0.500
Saturatio 1.000
Value 1.000
Fac 1.000

Inputs

Factor
: The amount of influence the node exerts on the image.

Image/Color
: Standard color input.

Hue
: The hue rotation offset, from 0 (-180°) to 1 (+180°). Note that 0 and 1 have the same result.

Saturation
: A value of 0 removes color from the image, making it black-and-white. A value greater than 1.0 increases saturation.

Value
: The value shift. 0 makes the color black, 1 keeps it the same, and higher values make it brighter.

Outputs

Image/Color
Standard color output.

Hue/Saturation Tips

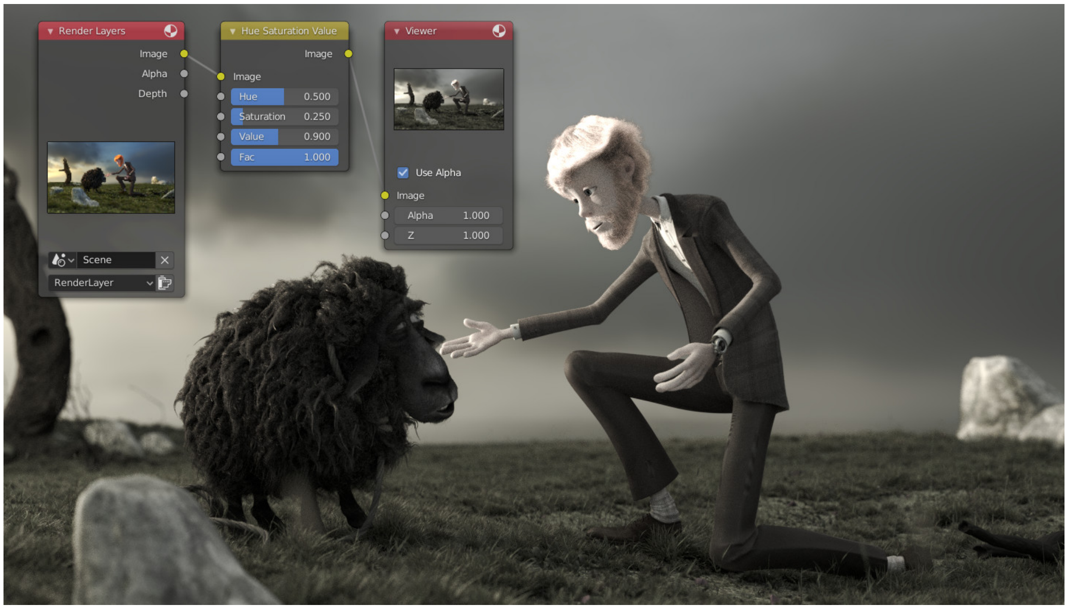
Some things to keep in mind that might help you use this node better:

Hues are laid out on a circle
If you apply a Hue offset of 1 (+180°) to a blue image, you get the diametrically opposite color, which is yellow. If you apply a Hue offset of 1 to that yellow image, you get blue again.

Grayscale images have no hue
Trying to change the Hue or Saturation of a grayscale image has no effect. You can only brighten or darken it by adjusting the Value. To add color, use the Mix node instead.

Changing the effect over time
The different values can be animated using a *Time Curve* node or by setting keyframes.

HSV Example

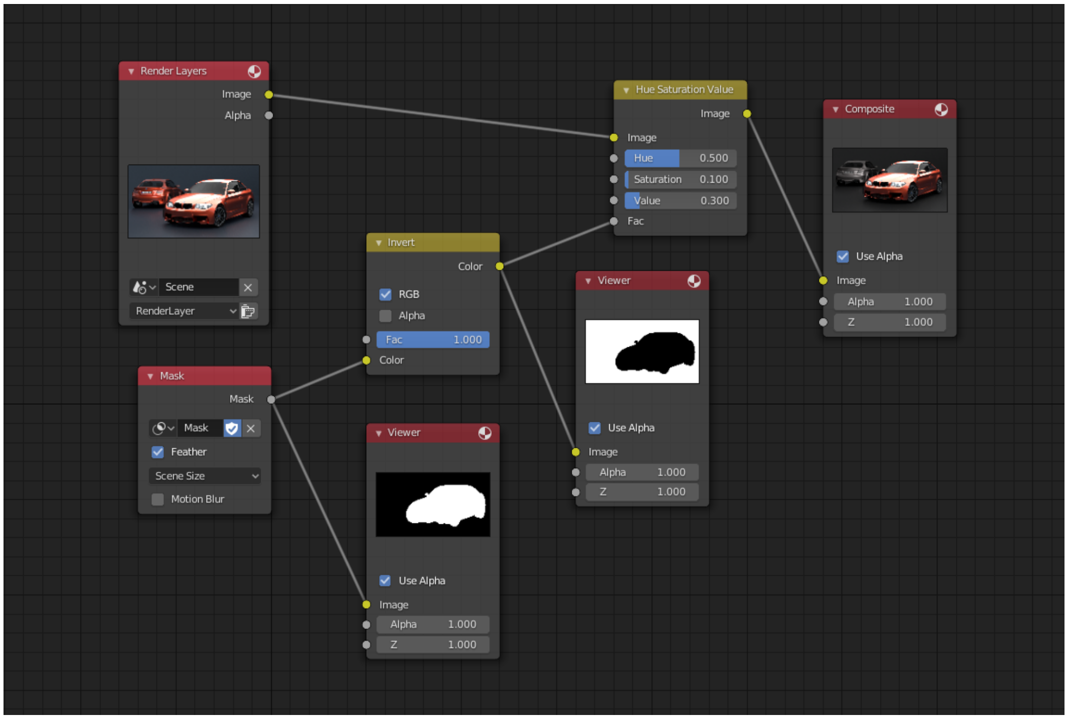


Scene
RenderLayer

Hue 0.500
Saturation 0.250
Value 0.900
Fac 1.000

☑ Use Alpha
Alpha 1.000
Z 1.000

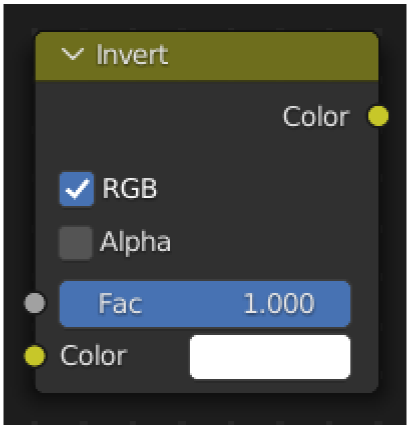
A basic example.


Render Layers
Image
Alpha
Scene
RenderLayer
Mask
Mask
Mask
Feather
Scene Size
Motion Blur
Invert
Color
RGB
Alpha
Fac 1.000
Color
Viewer
Use Alpha
Image
Alpha 1.000
Z 1.000
Hue Saturation Value
Image
Image
Hue 0.500
Saturation 0.100
Value 0.300
Fac
Viewer
Use Alpha
Image
Alpha 1.000
Z 1.000
Composite
Use Alpha
Image
Alpha 1.000
Z 1.000

An example of using the Factor input for masking.

Invert Color Node

Inverts the colors in the input image, producing a negative.



☑ RGB

☐ Alpha

Fac 1.000

Inputs

Factor
: The amount of influence the node exerts on the image.

Color
: Standard color input.

Properties

In the compositing context, this node has the following properties:

RGB
: Invert the color channels.

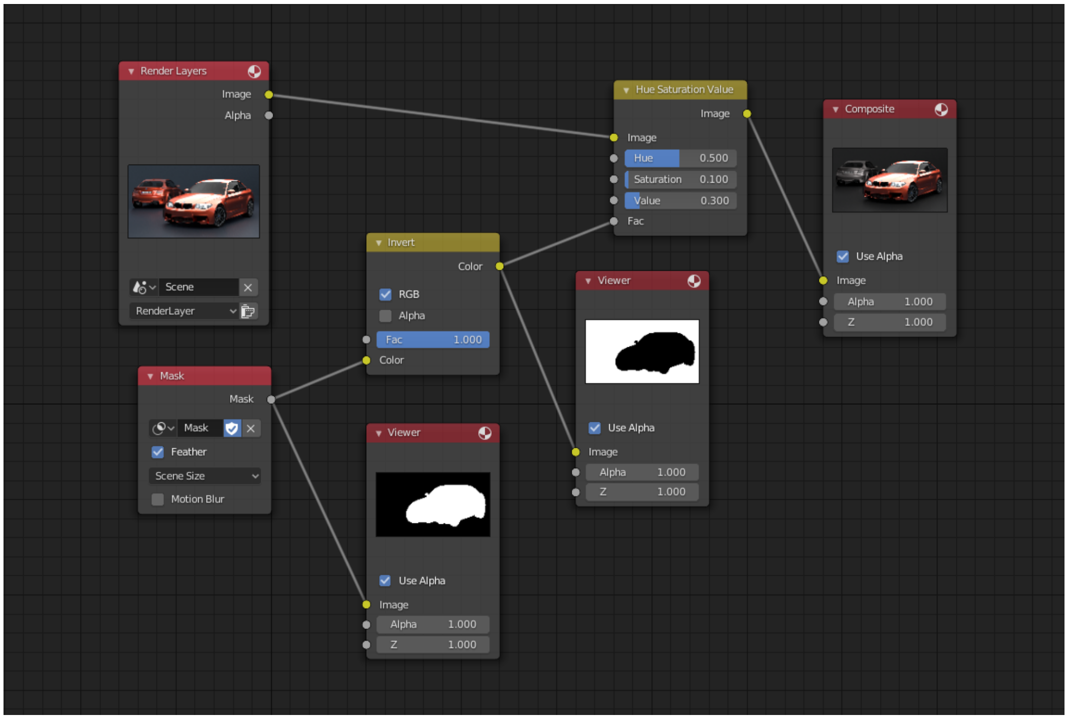
Alpha
: Invert the alpha channel.

Outputs

Color

Standard color output.

Example


Render Layers
Image
Alpha
Scene
RenderLayer
Hue Saturation Value
Image
Image
Fac
Composite
Image
Invert
Color
Color
Mask
Mask
Mask
Scene Size
Viewer
Image
Viewer
Image

Hue 0.500
Saturation 0.100
Value 0.300

☑ Use Alpha
Alpha 1.000
Z 1.000

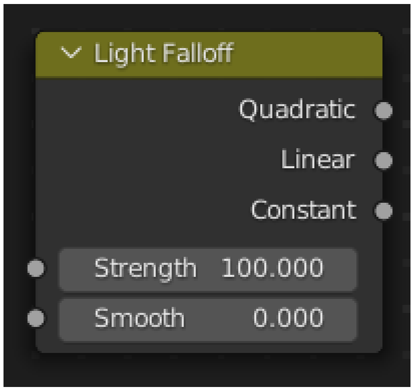
☑ RGB
☐ Alpha
Fac 1.000

☑ Feather
☐ Motion Blur

The Invert node is used to invert the mask.

Light Falloff Node

Cycles Only



The *Light Falloff* node allows you to manipulate how light intensity decreases over distance. In reality light will always fall off quadratically; however, it can be useful to manipulate as a non-physically-based lighting trick. Note that using Linear or Constant falloff may cause more light to be introduced with every global illumination bounce, making the resulting image extremely bright if many bounces are used.

Inputs

Strength
: Light strength before applying falloff modification.

Smooth
: Smooth intensity of light near light sources. This can avoid harsh highlights, and reduce global illumination noise. 0.0 corresponds to no smoothing; higher values smooth more. The maximum light strength will be strength/smooth.

Properties

This node has no properties.

Outputs

Quadratic

Quadratic light falloff; this will leave strength unmodified if smooth is 0.0 and corresponds to reality.

Linear

Linear light falloff, giving a slower decrease in intensity over distance.

Constant

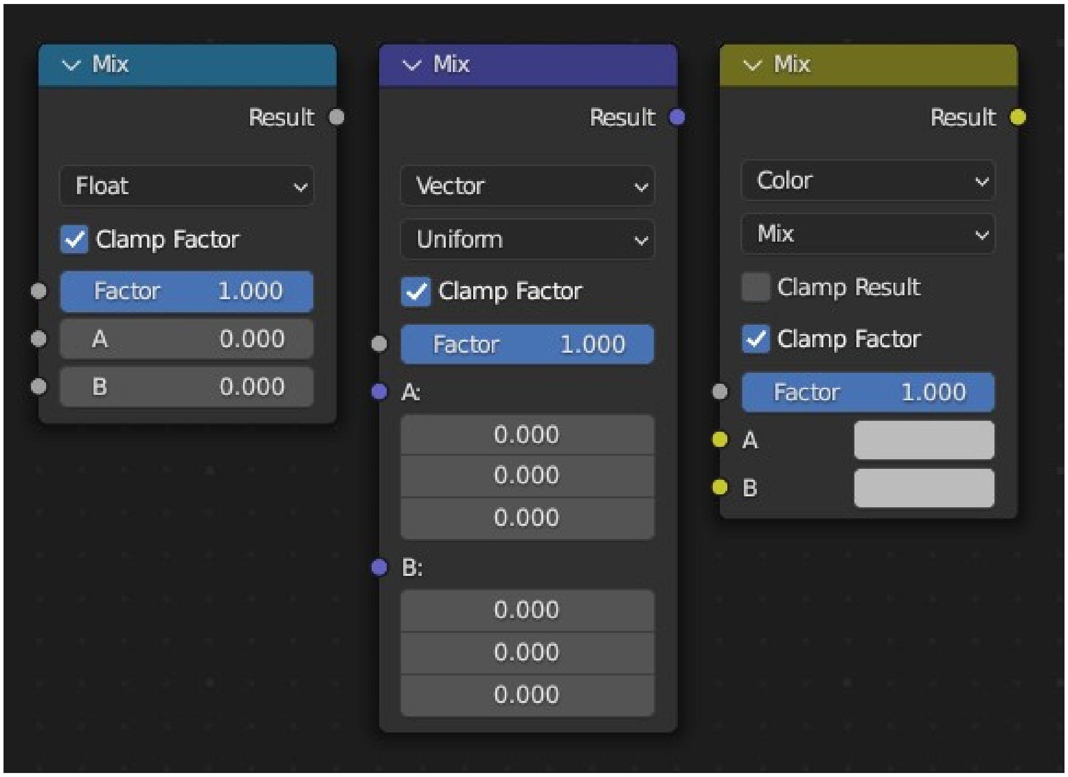
Constant light falloff, where the distance to the light has no influence on its intensity.

Examples

Todo <2.8 add.

Mix Color Node

The *Mix Node* mixes values, colors and vectors inputs using a factor to control the amount of interpolation. The *Color* mode has additional blending modes.


Mix
Result
Float
Clamp Factor
Factor 1.000
A 0.000
B 0.000
Mix
Result
Vector
Uniform
Clamp Factor
Factor 1.000
A:
0.000
0.000
0.000
B:
0.000
0.000
0.000
Mix
Result
Color
Mix
Clamp Result
Clamp Factor
Factor 1.000
A
B

Inputs

Factor
: Controls the amount of mixing between the A and B inputs.

A/B
: The two inputs that are mixed together.

Properties

Data Type
The data type that is used for mixing. The node supports float, vector, color, and rotation data types.

Factor Mode (Vector only)
The factor mode can be set to *Uniform* and *Non-Uniform*. In uniform mode, a single float controls the factor. In non-uniform mode, a vector controls the factor for each XYZ channel separately.

Mix (Color only)
The Blend modes can be selected in the select menu. See Color Blend Modes for details on each blending mode.

Add, Subtract, Multiply, Screen, Divide, Difference, Darken, Lighten, Overlay, Color Dodge, Color Burn, Hue, Saturation, Value, Color, Soft Light, Linear Light

Clamp Factor
Limit the factor value between 0.0 and 1.0. If this option is unchecked then the node operates using *Extrapolation*.

Clamp Result (Color only)
Limit the Result to the range between 0.0 and 1.0.

Outputs

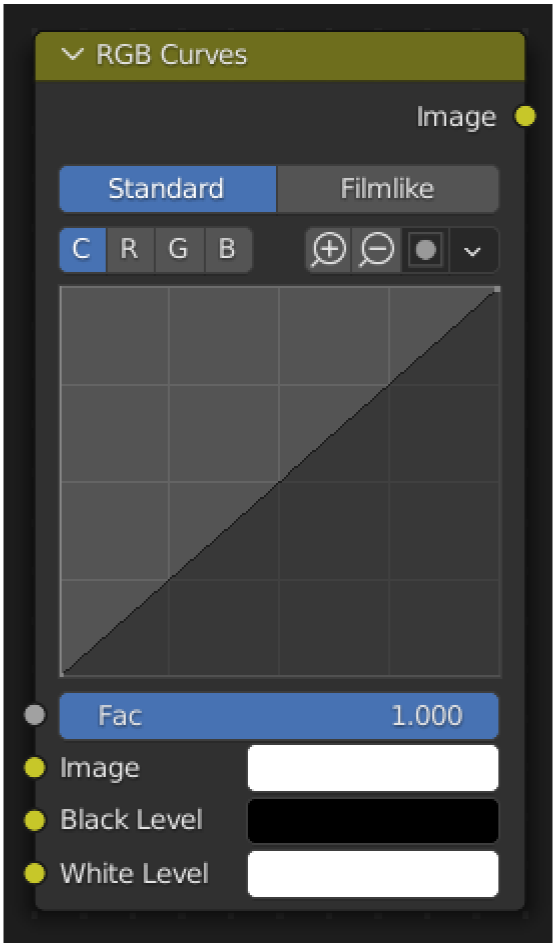
Result
Output the result of the mix using the data type selected.

Examples

See the Color > Mix page for additional examples: Mix Color Node

RGB Curves Node

The *RGB Curves Node* performs level adjustments on each color channel.


RGB Curves
Image
Standard
Filmlike
C
R
G
B
Fac
1.000
Image
Black Level
White Level

Inputs

Factor
: Controls the amount of influence the node exerts on the image.

Image/Color
: Standard color input.

Black Level *Compositor Only*
: Defines the input color that should be mapped to black.

White Level *Compositor Only*

Defines the input color that should be mapped to white.

Tip

To define the black and white levels, use the eyedropper to select a color sample of a displayed image.

Properties

Tone *Compositor Only*

Standard::	The Combined curve is applied to each channel individually, which may result in a change of hue.
Filmlike::	Keeps the hue constant.

Channel

The curve to show.

C::	Combined
R::	Red
G::	Green
B::	Blue

Curve

A Bézier curve that maps each input level (X axis) to an output level (Y axis). For the curve controls, see Curve widget.

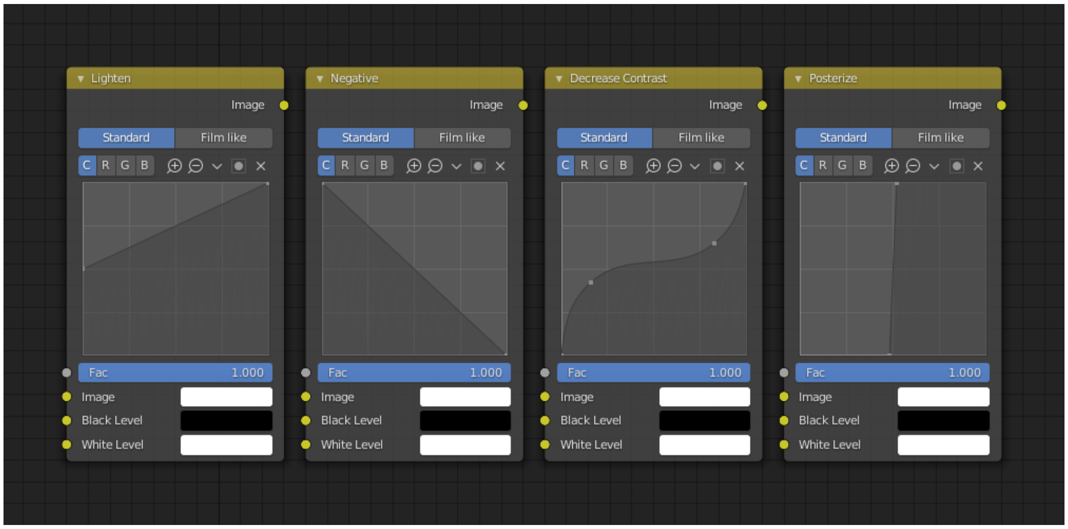
Outputs

Image/Color

Standard color output.

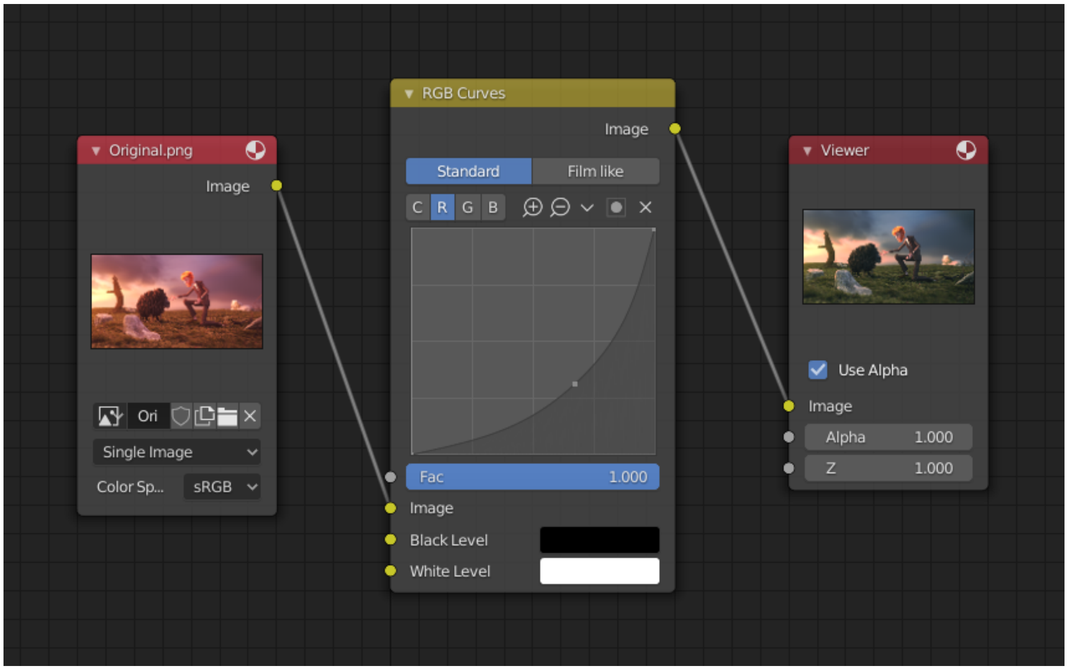
Examples

Below are some common curves you can use to achieve desired effects.


Lighten
Negative
Decrease Contrast
Posterize
Image
Standard
Film like
C R G B
Fac 1.000
Image
Black Level
White Level

From left to right: 1. Lighten shadows 2. Negative 3. Decrease contrast 4. Posterize.

Color Correction using Curves

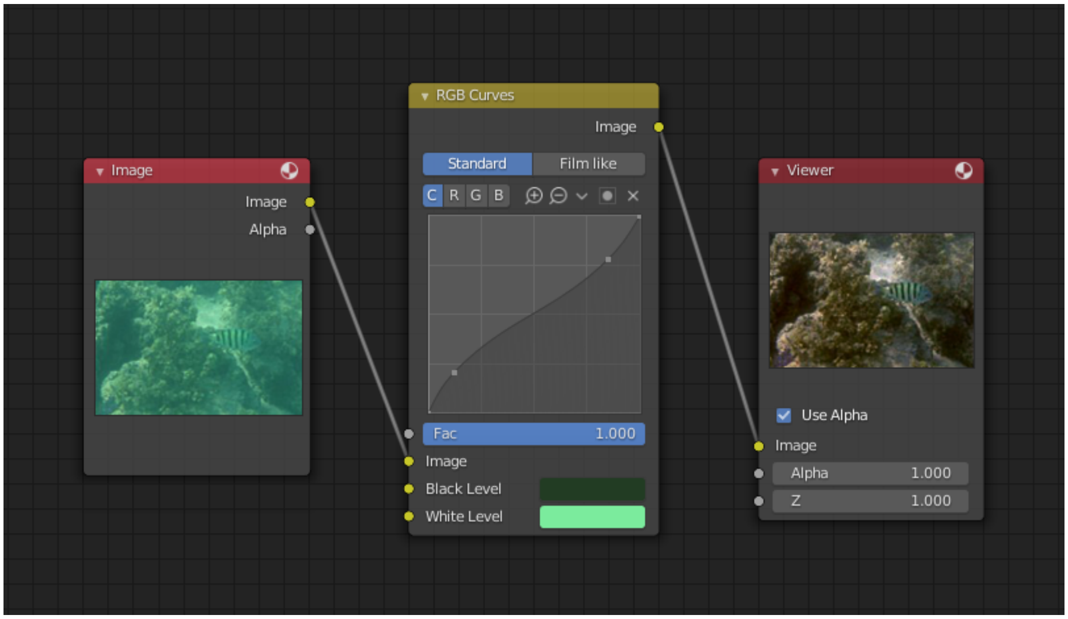

Original.png
Image
Ori
Single Image
Color Sp...
sRGB
RGB Curves
Image
Standard
Film like
C R G B
Fac 1.000
Image
Black Level
White Level
Viewer
Use Alpha
Image
Alpha 1.000
Z 1.000

Color correction with curves.

In this example, the image has too much red in it, so we run it through an *RGB Curves* node and reduce the Red channel.

The documentation for the Mix Node has an additional example about fixing overexposure.

Color Correction using Black/White Levels



Fac 1.000

☑ Use Alpha

Alpha 1.000

Z 1.000

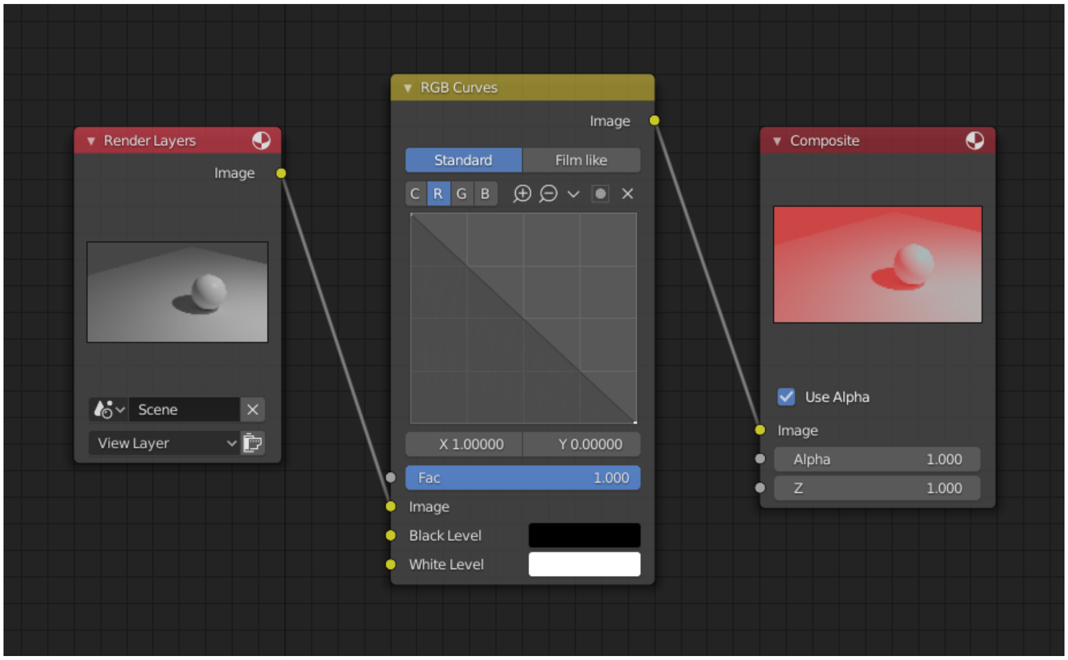
Color correction with Black/White Levels.

Manually adjusting the RGB curves for color correction can be difficult. Another option for color correction is to use the Black and White Levels instead, which really might be their main purpose.

In this example, the White Level is set to the color of a bright spot of the sand in the background, and the Black Level to the color in the center of the fish’s eye. To do this efficiently it is best to bring up the Image Editor showing the original input image. You can then use the levels’ color picker to easily choose the appropriate colors from the input image, zooming into pixel level if necessary. The result can be fine-tuned with the R, G, and B curves like in the previous example.

The curve for C is used to compensate for the increased contrast that is a side effect of setting Black and White Levels.

Effects


Render Layers
Image
Scene
View Layer
RGB Curves
Image
Standard
Film like
C
R
G
B
Image
Black Level
White Level
Composite
Image

X 1.00000 Y 0.00000

Fac 1.000

☑ Use Alpha

Alpha 1.000

Z 1.000

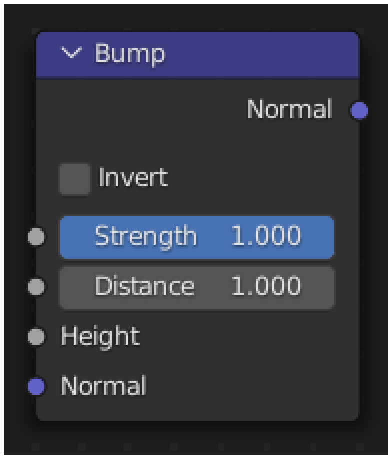
Changing colors by inverting the red channel.

Vector

- Bump Node
- Displacement Node
- Mapping Node
- Normal Node
- Normal Map Node
- Vector Curves Node
- Vector Displacement Node
- Vector Rotate Node
- Vector Transform Node

Bump Node

The *Bump* node generates a perturbed normal from a height texture, for bump mapping. The height value will be sampled at the shading point and two nearby points on the surface to determine the local direction of the normal.



Inputs

Strength
: Strength of the bump mapping effect, interpolating between no bump mapping and full bump mapping.

Distance
: Multiplier for the height value to control the overall distance for bump mapping.

Height
: Scalar value giving the height offset from the surface at the shading point; this is where you plug in textures.

Normal
: Standard normal input.

Properties

Invert
: Invert the bump mapping, to displace into the surface instead of out.

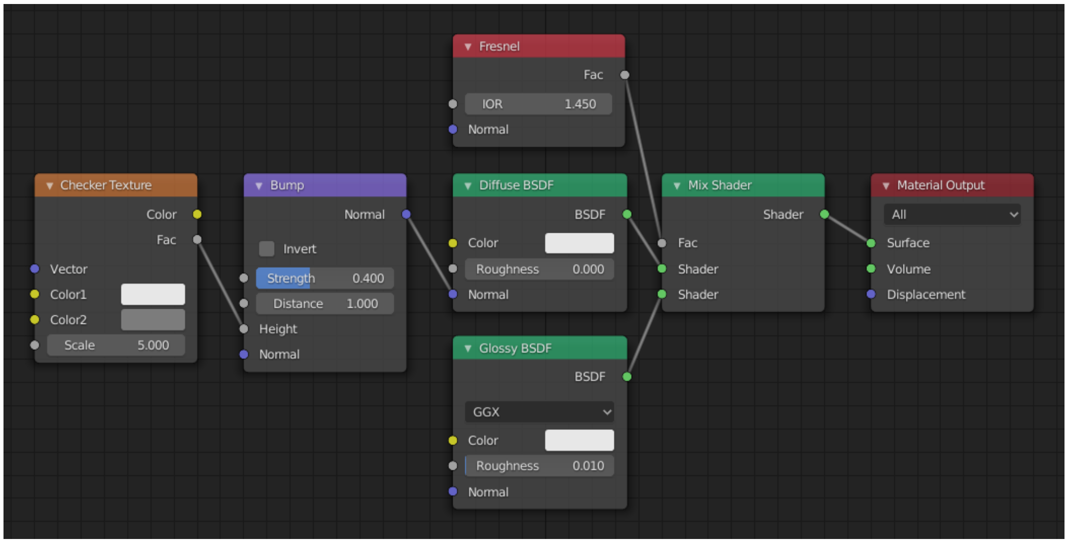
Outputs

Normal
: Standard normal output.

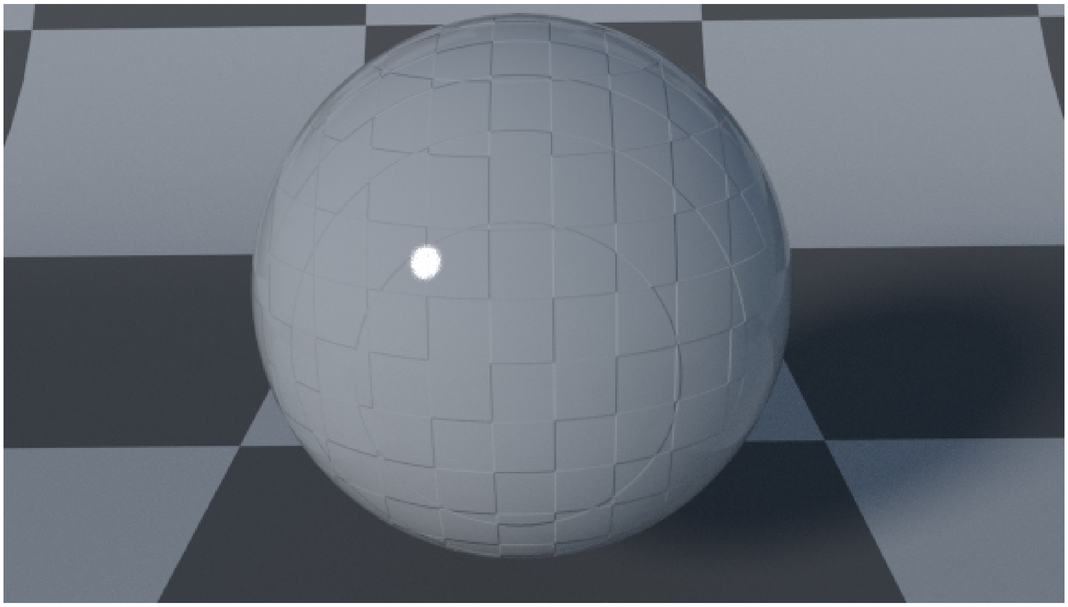
Tip

If the *Height* input is not connected, the node becomes a no-op that outputs its *Normal* input as is, or defaults to the geometry normal if not connected. Routing a node group input via a no-op Bump node before doing math effectively makes it default to normal.

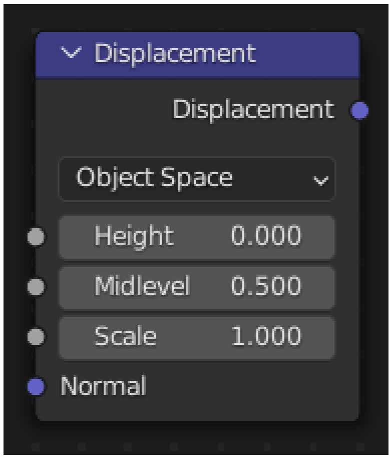
Examples


Fresnel
Fac
IOR 1.450
Normal
Checker Texture
Color
Fac
Vector
Color1
Color2
Scale 5.000
Bump
Normal
Invert
Strength 0.400
Distance 1.000
Height
Normal
Diffuse BSDF
BSDF
Color
Roughness 0.000
Normal
Mix Shader
Shader
Fac
Shader
Shader
Material Output
All
Surface
Volume
Displacement
Glossy BSDF
BSDF
GGX
Color
Roughness 0.010
Normal

The above node setup will only bump the diffuse part of the shader, simulating a bumpy diffuse surface coated with a smooth glossy “glaze” layer.



Displacement Node



Displacement

Displacement

Object Space

Height 0.000

Midlevel 0.500

Scale 1.000

Normal

The *Displacement* node is used to displace the surface along the surface normal, to add more detail to the geometry. Both procedural textures and baked displacement maps can be used.

By default, Blender only uses Bump Mapping to render displacement. However with true displacement, the rendered geometry will be physically displaced. To use true displacement the Displacement method must be set accordingly.

Tip

For best results when using true displacement, the mesh must be subdivided finely to bring out the detail in the displacement texture.

See also

Material Displacement for more details on displacement workflows.

Inputs

Height
: Distance to displace the surface along the normal. This is where a texture node can be connected.

Midlevel
: Neutral displacement value that causes no displacement. With the default 0.5, any lower values will cause the surfaces to be pushed

inwards, and any higher values will push them outwards.

Scale
: Increase or decrease the amount of displacement.

Normal
: Standard normal input.

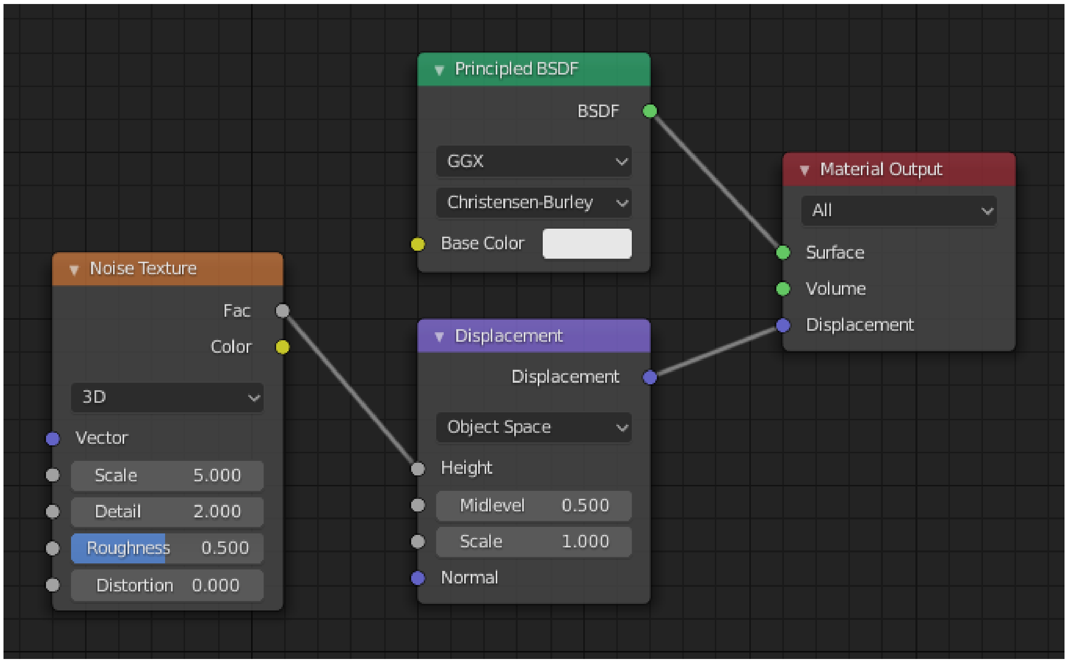
Properties

Space
: Object Space means the displacement scales along with the object. When using World Space the object scale is ignored.

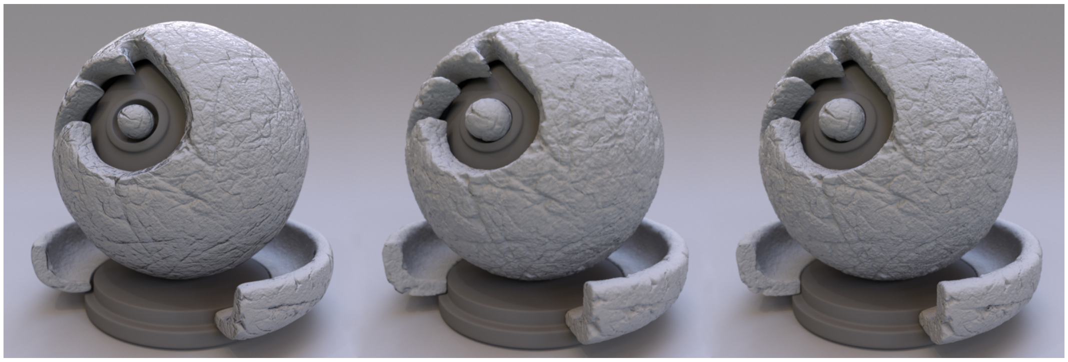
Outputs

Displacement
: Displacement offset to be connected into the Material Output.

Examples


Principled BSDF
BSDF
GGX
Christensen-Burley
Base Color
Material Output
All
Surface
Volume
Displacement
Noise Texture
Fac
Color
3D
Vector
Scale 5.000
Detail 2.000
Roughness 0.500
Distortion 0.000
Displacement
Displacement
Object Space
Height
Midlevel 0.500
Scale 1.000
Normal

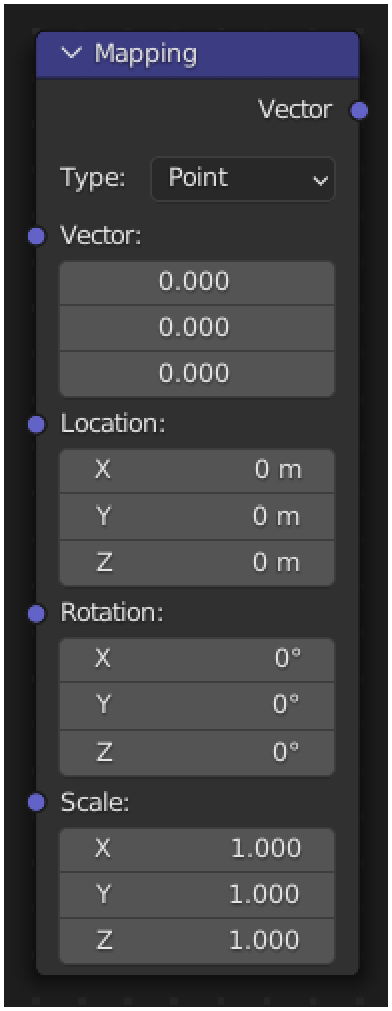
Typical displacement node setup.



Bump only, displacement only, and displacement and bump combined.

Mapping Node

The *Mapping* node transforms the input vector by applying translation, rotation, and scaling.



Type: Point

Vector:
0.000
0.000
0.000

Location:
X 0 m
Y 0 m
Z 0 m

Rotation:
X 0°
Y 0°
Z 0°

Scale:
X 1.000
Y 1.000
Z 1.000

Inputs

The inputs of the node are dynamic. In particular, the *Location* input is only available in the *Texture* and *Point* vector types.

Vector

The vector to be transformed.

Location

The amount of translation along each axis.

Rotation

The amount of rotation along each axis. XYZ order.

Scale

The amount of scaling along each axis.

Properties

Vector Type

The node applies the transformation differently depending on the semantic type of the input vector.

Point:: For this vector type, the node performs a straightforward transformation.

Transforming a texture coordinates is analogous to transforming a UV map. For instance, translating the texture coordinates along the **positive** X axis would result in the evaluated texture to move in the **negative** X axis, much like if one translated a UV map. Similarly, scaling the texture coordinates **up** would result in the evaluated texture to scale **down**. So transforming the texture coordinates would appear to have the opposite effect on the evaluated texture.

The order of transformation is: Scale –> Rotate –> Translate, which means:

- Translation moves the input along the local rotation axis.
- Rotation rotates the input around the origin of the space.
- Scaling scales the input along the global axis.

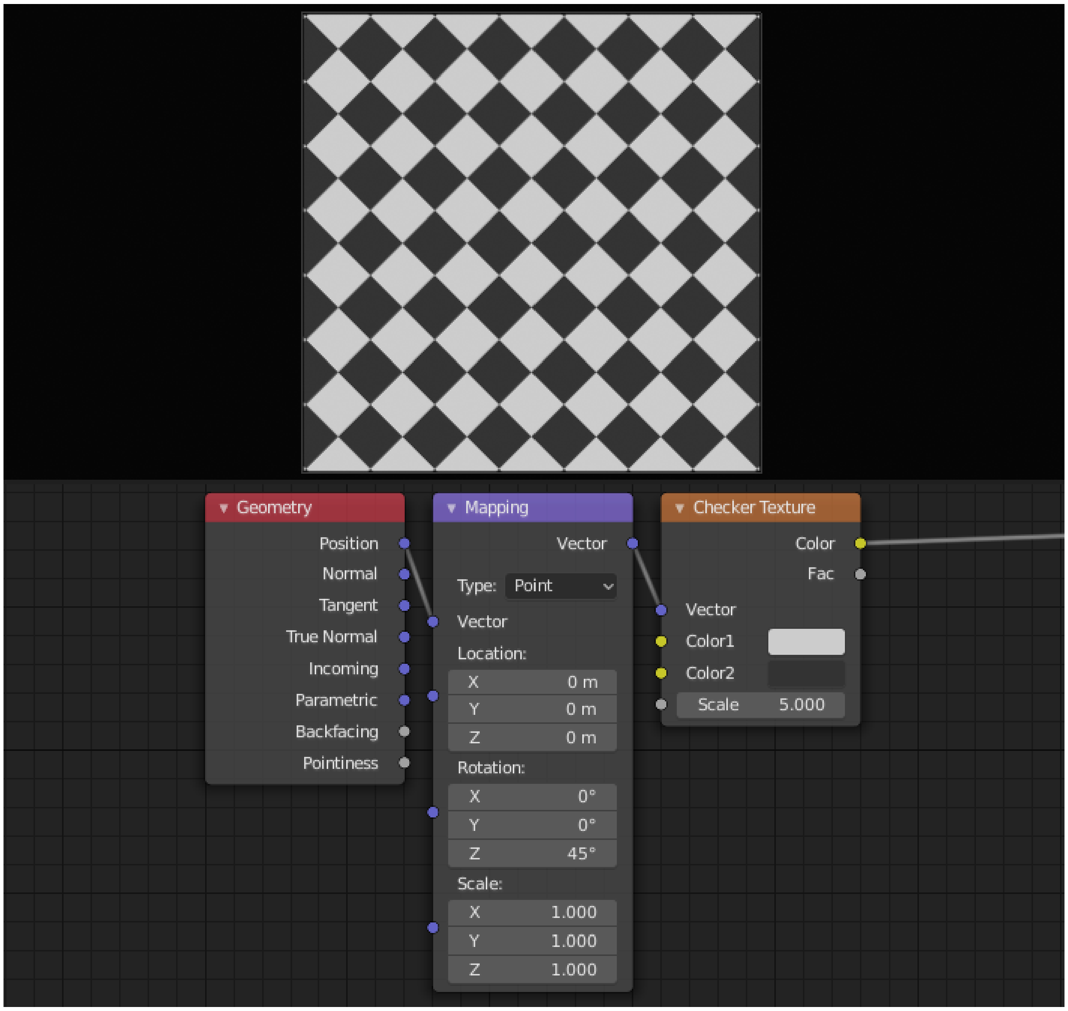
Texture::	For this vector type, the node performs an inverse transformation. Inverse transforming a texture coordinates would, as opposed to the *Point* type, transform the evaluated texture itself. For instance, translating the texture coordinates along the positive X axis would result in the evaluated texture to move in the positive X axis, as one would expected. Similarly, scaling the texture coordinates up would result in the evaluated texture to scale up, as one would expect. The order of transformation is: Translate –> Rotate –> Scale, which means: • Translation moves the input along the global axis. • Rotation rotates the input around the translation vector. • Scaling scales the input along the local rotation axis.
Vector::	For this vector type, a *Point* transformation is performed, but with zero translation.
Normal::	For this vector type, the node performs the inverse transpose of the transformation and normalize the result. Such transformation ensures correct normals after non-uniform scaling. So this type should be used when transforming normals.

Outputs

Vector

The input vector after transformation.

Examples


Geometry
Position
Normal
Tangent
True Normal
Incoming
Parametric
Backfacing
Pointiness
Mapping
Vector
Vector
Checker Texture
Color
Fac
Vector
Color1
Color2

Type: Point

Location:

X 0 m

Y 0 m

Z 0 m

Rotation:

X 0°

Y 0°

Z 45°

Scale:

X 1.000

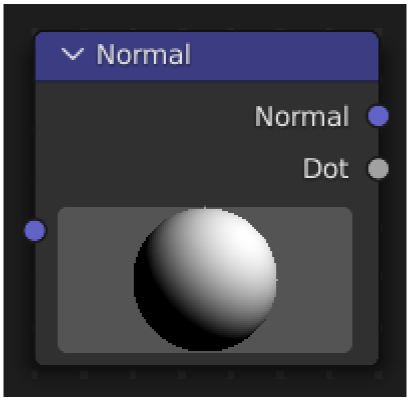
Y 1.000

Z 1.000

Scale 5.000

Mapping node example.

Normal Node



The Normal node generates a normal vector and a dot product.

Inputs

Normal
: Normal vector input.

Properties

Normal Direction
: To manually set a fixed normal direction vector. `LMB` click and drag on the sphere to set the direction of the normal. Holding `Ctrl` while dragging snaps to 45 degree rotation increments.

Outputs

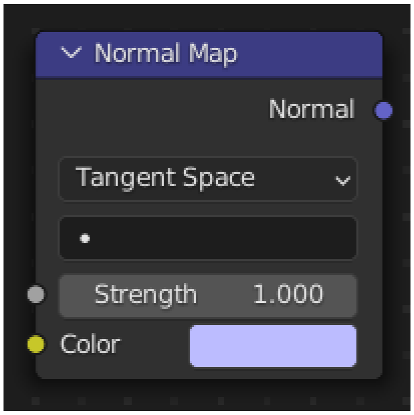
Normal
: Normal vector output.

Dot
: Dot product output. The dot product is a scalar value.

  - If two normals are pointing in the same direction the dot product is 1.

- If they are perpendicular the dot product is zero (0).
- If they are antiparallel (facing directly away from each other) the dot product is -1.

Normal Map Node



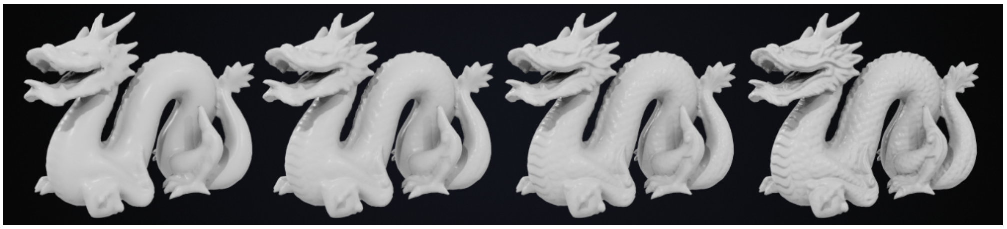
Tangent Space

Strength 1.000

The *Normal Map* node generates a perturbed normal from an RGB normal map image. This is usually chained with an *Image Texture* node in the color input, to specify the normal map image. For tangent space normal maps, the UV coordinates for the image must match, and the image texture should be set to *Non-Color* mode to give correct results.

Inputs

Strength
: Strength of the normal mapping effect.



Strength is set to 0, 0.5, 1, 2 (from left to right).

Color
: RGB color that encodes the normal map in the specified space.

Properties

Space
: The input RGB color can be in one of three spaces: Tangent, Object and World space. Tangent space normal maps are the most common, as they support object transformation and mesh deformations. Object space

normal maps keep sticking to the surface under object transformations, while World normal maps do not.

UV Map

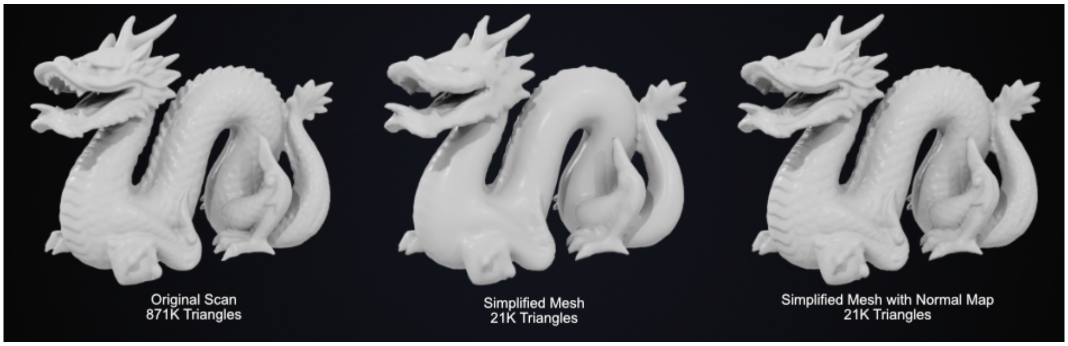
Name of the UV map to derive normal mapping tangents from. When chained with an Image Texture node, this UV map should be the same as the UV map used to map the texture.

Outputs

Normal

Normal that can be used as an input to BSDF nodes.

Example

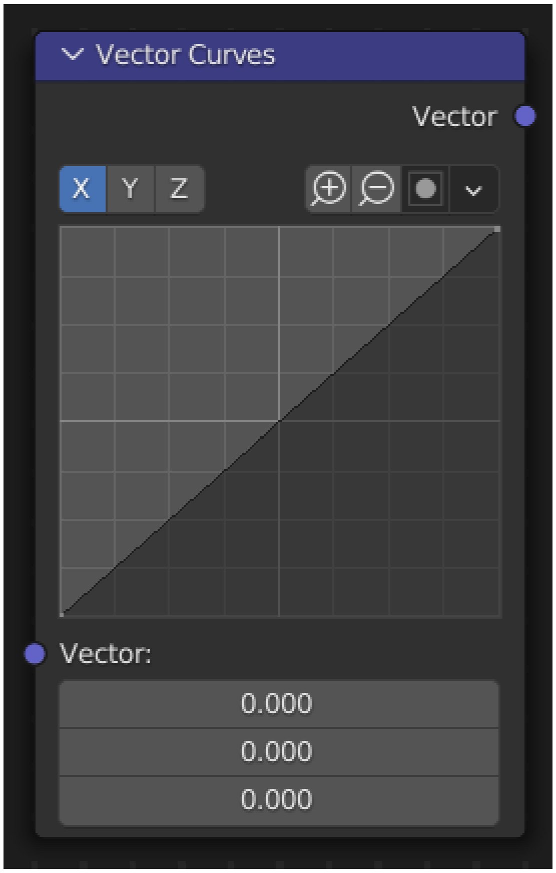

Original Scan
871K Triangles
Simplified Mesh
21K Triangles
Simplified Mesh with Normal Map
21K Triangles

The Normal Map Strength is set to 1.

Vector Curves Node

The Vector Curves node maps an input vector components to a curve.

Use this curve node to slow things down or speed them up from the original scene.



Vector:
0.000
0.000
0.000

Inputs

In the shader context the node also has an additional Factor property.

Factor

    Controls the amount of influence the node exerts on the output vector.

Vector

    Standard vector input.

Properties

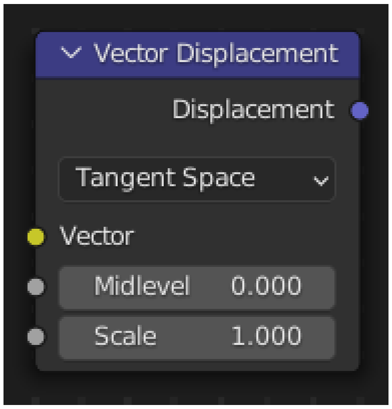
Channel
X, Y, Z

Curve
For the curve controls see: Curve widget.

Outputs

Vector
Standard vector output.

Vector Displacement Node



Tangent Space

Midlevel 0.000

Scale 1.000

The *Vector Displacement* node is used to displace the surface along arbitrary directions, unlike the regular Displacement node which only displaces along the surface normal.

It is typically used to apply vector displacement maps created by other sculpting software. Vector displacement maps can fully represent the high resolution detail to be applied on a smooth base mesh, unlike regular displacement maps.

By default, Blender only uses Bump Mapping to render displacement. However with true displacement, the rendered geometry will be physically displaced. To use true displacement the Displacement method must be set accordingly.

Tip

For best results when using true displacement, the mesh must be subdivided finely to bring out the detail in the displacement texture.

See also

Material Displacement for more details on displacement workflows.

Inputs

Vector
: Vector specifying the displacement along three axes. This is where a texture node can be connected.

Typically a baked vector displacement image texture is used. For Object Space, RGB colors in the image are interpreted as an XYZ offset in object space. For Tangent Space, R is an offset along the tangent, G along the normal and B along the bitangent.

Midlevel

Neutral displacement value that causes no displacement. With the default 0.0, any lower values will cause the surfaces to be pushed inwards, and any higher values will push them outwards.

Scale

Increase or decrease the amount of displacement.

Properties

Space

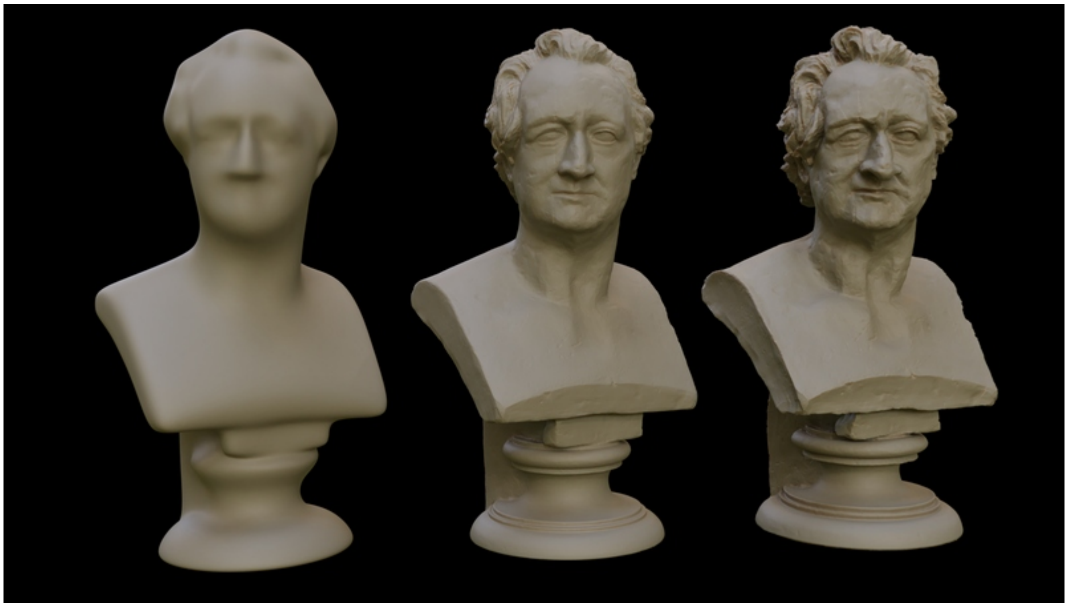
Object Space maps work for static meshes, and will render slightly faster with less memory usage. Tangent Space maps can be used for meshes that will be deformed, like animated characters, so the displacement follows the deformation.

Outputs

Displacement

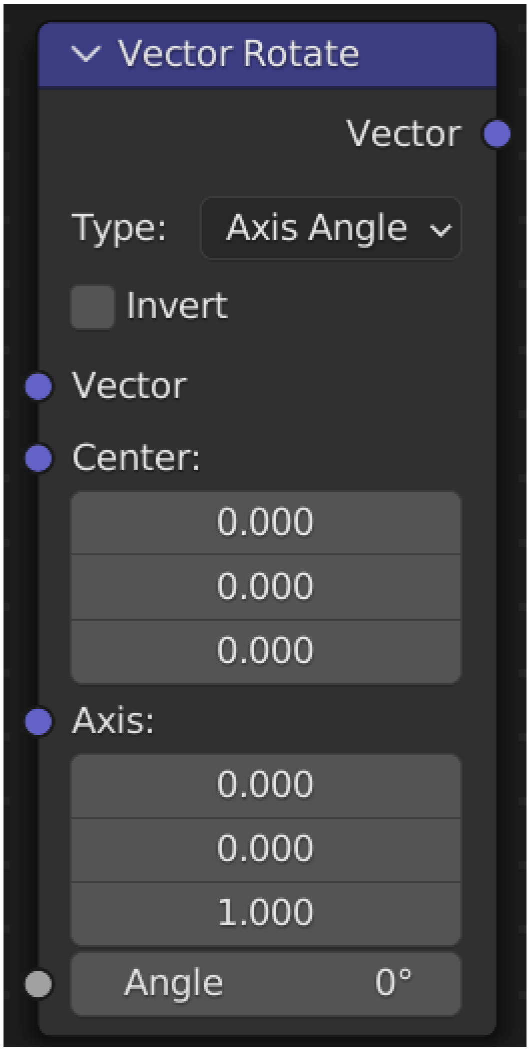
Displacement offset to be connected into the Material Output.

Examples



Regular and exaggerated vector displacement on a smooth base mesh.

Vector Rotate Node


Vector Rotate

Vector

Type: Axis Angle

☐ Invert

Vector

Center:

0.000

0.000

0.000

Axis:

0.000

0.000

1.000

Angle 0°

The *Vector Rotate Node* provides the ability to rotate a vector around a pivot point (*Center*).

Inputs

Vector
: Vector to be rotated.

Center

Point to rotate around.

Axis

Axis to rotate around.

Angle

Angle to rotate the input vector by.

Rotation

When *Type* is set to *Euler*, rotate the input vector by these angles around the X, Y, then Z axes in that order.

Properties

Type

The type of angle input.

X/Y/Z Axis::	Rotates the vector around the defined axis and the amount of rotation is defined by the *Angle* input.
Axis Angle::	Rotates the vector around an arbitrary axis defined by the *Axis* input vector. The amount of rotation is defined by the *Angle* input.
Euler::	Rotates the vector about a center point defined by the *Center* input vector. The amount of rotation on each axis is defined by the *Rotation* input vector.

Invert

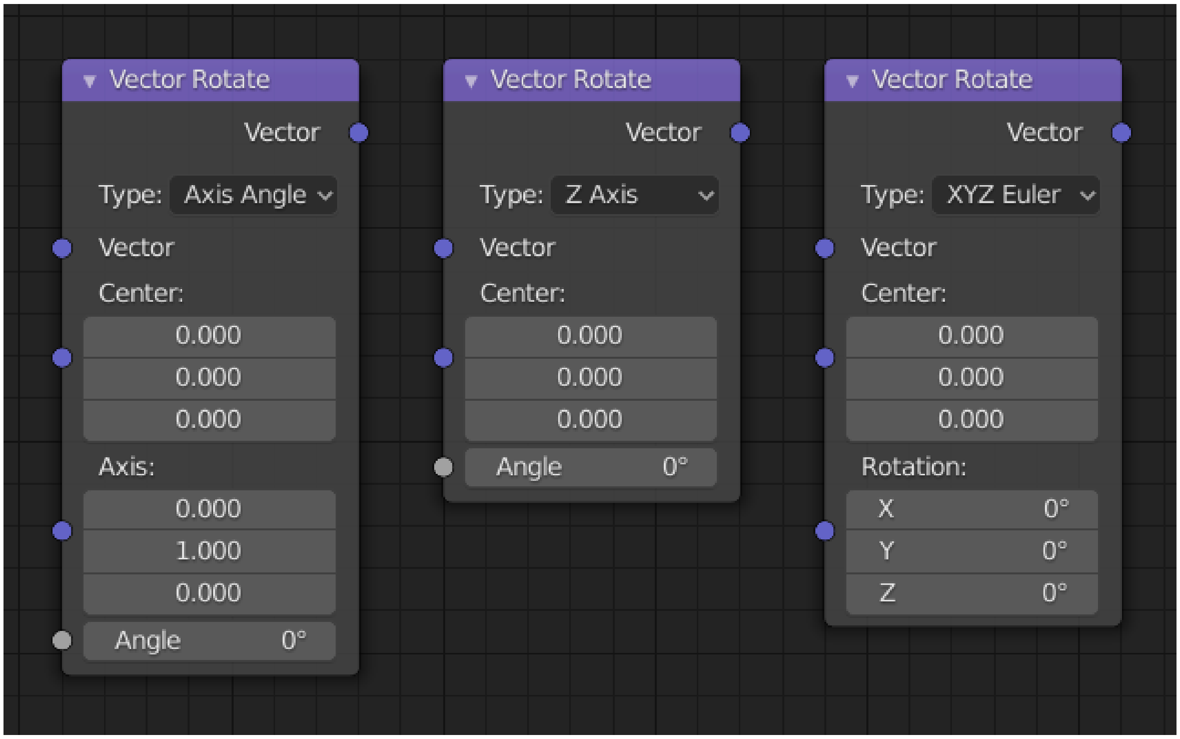
Inverts the rotation angle.

Outputs

Vector

The rotated vector.

Examples



Vector Rotate
Vector
Type: Axis Angle
Vector
Center:
0.000
0.000
0.000
Axis:
0.000
1.000
0.000
Angle 0°

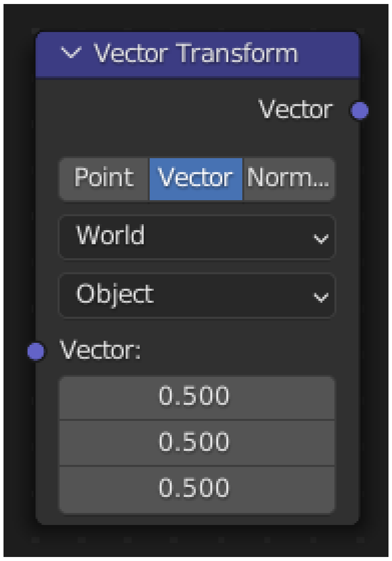
Vector Rotate
Vector
Type: Z Axis
Vector
Center:
0.000
0.000
0.000
Angle 0°

Vector Rotate
Vector
Type: XYZ Euler
Vector
Center:
0.000
0.000
0.000
Rotation:
X 0°
Y 0°
Z 0°

Vector Rotate node example.

Vector Transform Node

The *Vector Transform* node allows converting a vector, point, or normal between world and camera and object coordinate space.


Vector Transform
Vector
Point
Vector
Norm...
World
Object
Vector:
0.500
0.500
0.500

Inputs

Vector Input

Standard vector input.

Properties

Type

Specifies the input/output type.

Vector, Point, Normal.

Convert From

Coordinate Space to convert from:

World, Object, Camera.

Convert To

Coordinate Space to convert to:

World, Object, Camera.

Outputs

Vector Output
The transformed output vector.

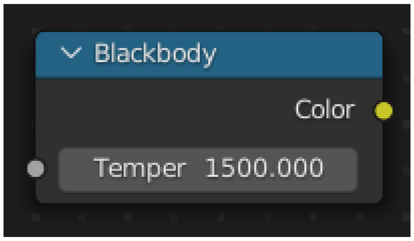
Examples

Todo <2.8 add.

Converter

- Blackbody Node
- Clamp Node
- Color Ramp Node
- Combine Color Node
- Combine XYZ Node
- Float Curve
- Map Range Node
- Math Node
- Mix Node
- RGB to BW Node
- Separate Color Node
- Separate XYZ Node
- Shader To RGB
- Vector Math Node
- Wavelength Node

Blackbody Node


Blackbody
Color
Temper 1500.000

The *Blackbody* node converts a blackbody temperature to RGB value. This can be useful for materials that emit light at natural occurring frequencies.

Inputs

Temperature
    The temperature in Kelvin.

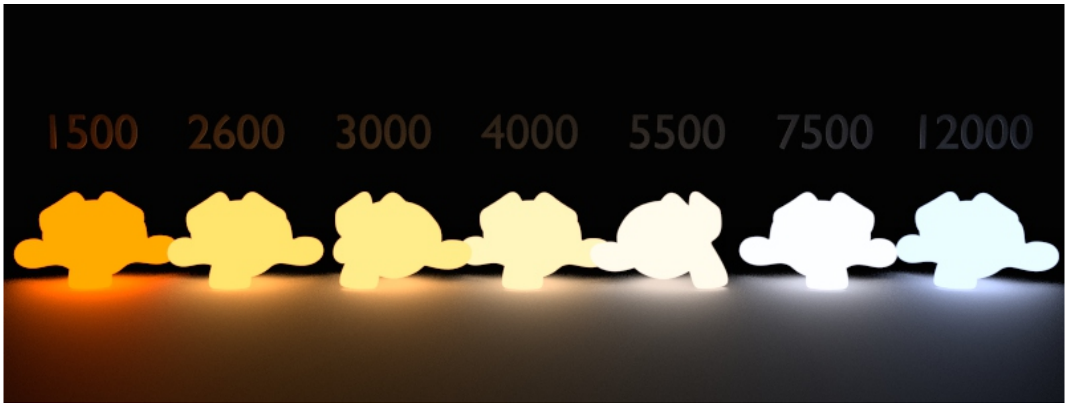
Properties

This node has no properties.

Outputs

Color
    RGB color output.

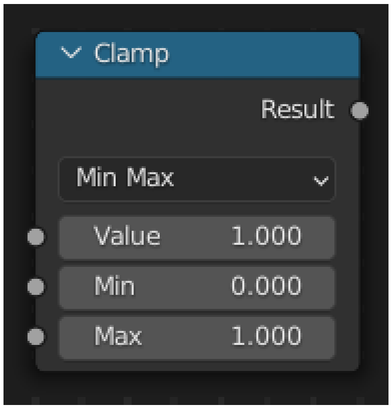
Examples


1500
2600
3000
4000
5500
7500
12000

Example of the color ranges of the Blackbody node.

Clamp Node

The *Clamp* node clamps a value between a minimum and a maximum.



Min Max

Value 1.000

Min 0.000

Max 1.000

Inputs

Value
    The input value to be clamped.

Min
    The minimum value.

Max
    The maximum value.

Properties

Clamp Type
    Method to clamp.

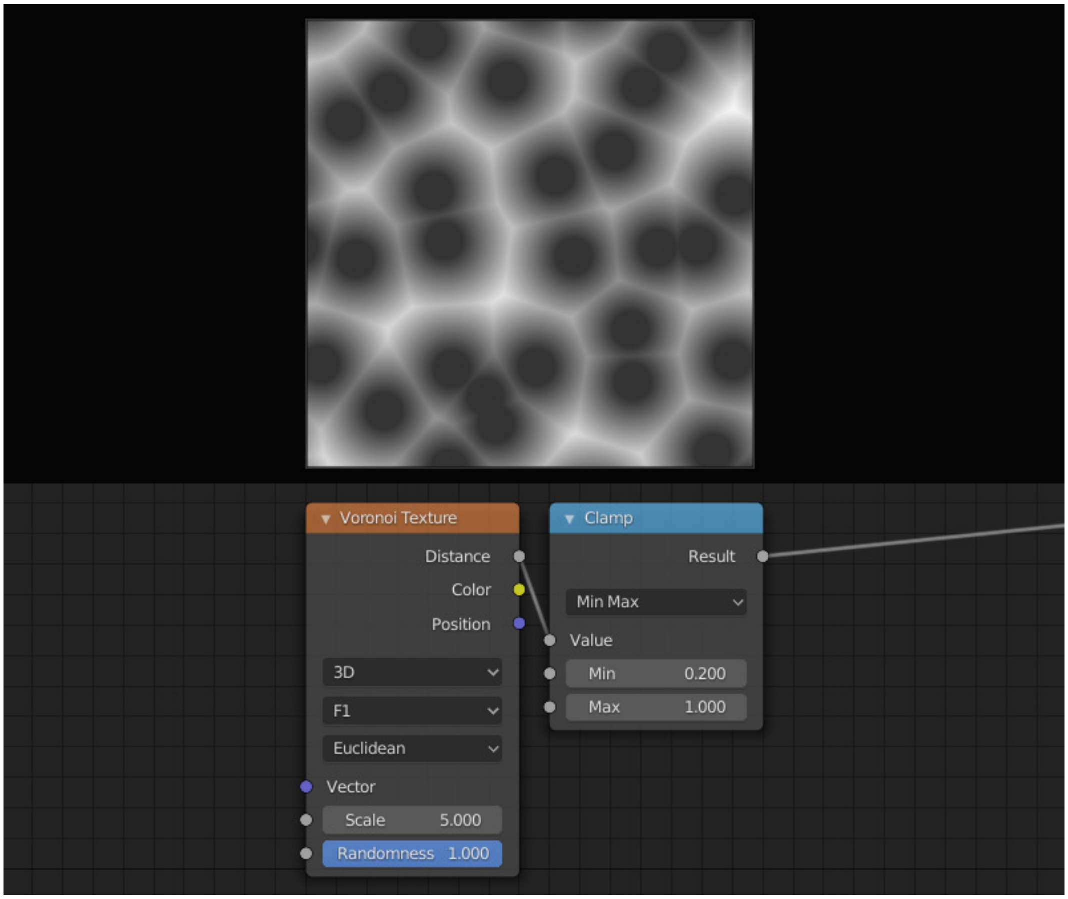
Min Max::	Constrain values between Min and Max.
Range::	Constrain values between Min and Max. When Min is greater than Max, constrain between Max and Min instead.

Outputs

Result
The input value after clamping.

Examples

The *Voronoi Texture* node outputs a value whose minimum is zero. We can use the *Clamp* node to clamp this value such that the minimum is 0.2.

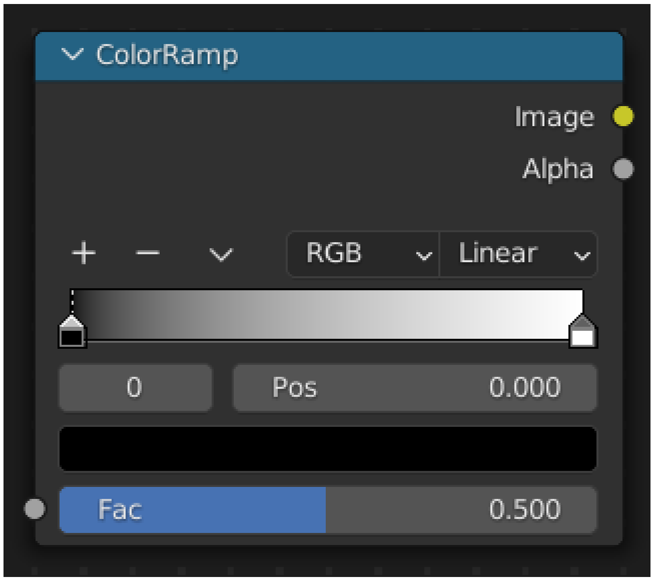

Voronoi Texture
Distance
Color
Position
Vector
Clamp
Result
Value

3D
F1
Euclidean
Scale 5.000
Randomness 1.000
Min Max
Min 0.200
Max 1.000

Example of Clamp node.

Color Ramp Node

The Color Ramp Node is used for mapping values to colors using a gradient.



Pos 0.000

Fac 0.500

Inputs

Factor

The value to map. 0.0 results in the leftmost color, while 1.0 results in the rightmost.

Properties

Color Ramp

See Color Ramp Widget.

Outputs

Image/Color

Standard color output.

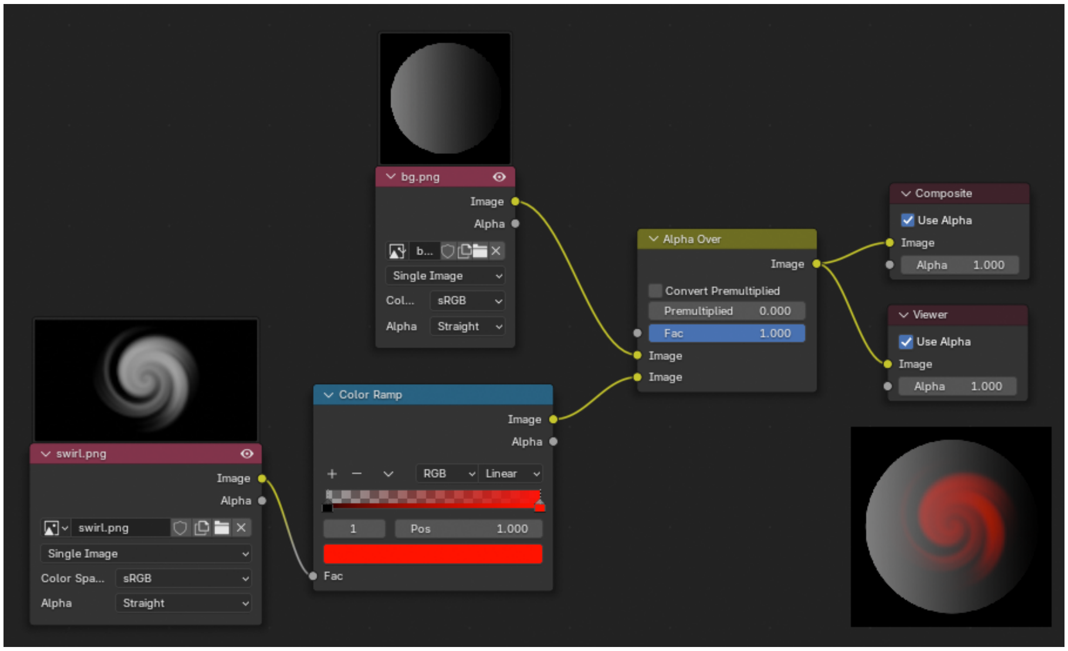
Alpha

Standard alpha output.

Examples

Creating an Alpha Mask

An often overlooked use case of the Color Ramp is to turn a black-and-white image into a colored image with transparency.

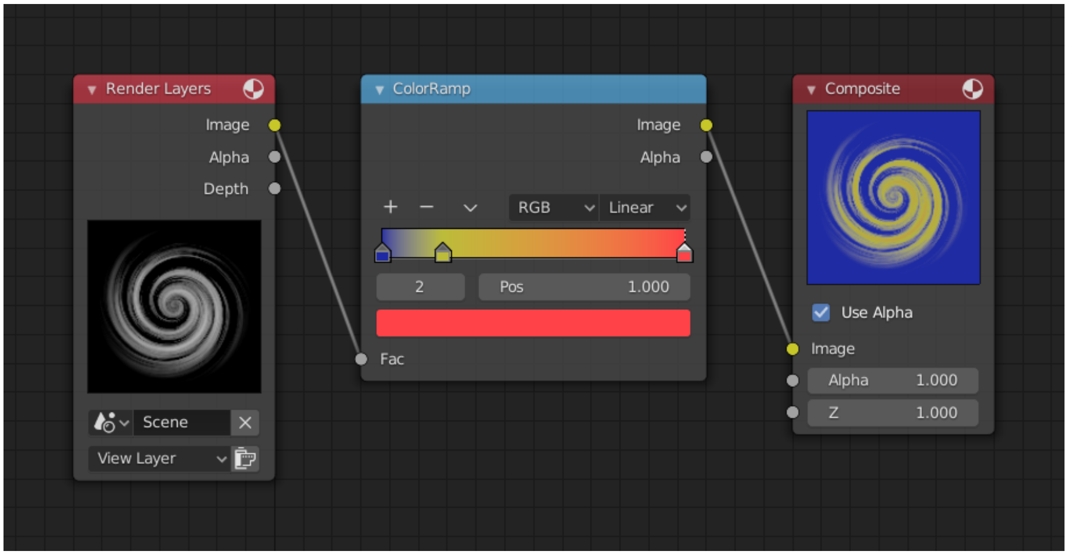

bg.png
Image
Alpha
b...
Single Image
Col... sRGB
Alpha Straight
swirl.png
Image
Alpha
swirl.png
Single Image
Color Spa... sRGB
Alpha Straight
Color Ramp
Image
Alpha
RGB
Linear
1
Pos 1.000
Fac
Alpha Over
Image
Convert Premultiplied
Premultiplied 0.000
Fac 1.000
Image
Image
Composite
Use Alpha
Image
Alpha 1.000
Viewer
Use Alpha
Image
Alpha 1.000

In the example above, a black-and-white swirl image, which is lacking an alpha channel, is fed into the Color Ramp node as a *Factor*.

The Color Ramp node is set to a purely transparent color on the left end of the gradient, and a fully red color on the right. As you can see in the Viewer node, the Color Ramp node outputs an image that is transparent where the input is black, and opaque where the input is white.

Colorizing an Image

In this example, multiple colors are added to the color gradient, converting a black-and-white image into a flaming swirl.

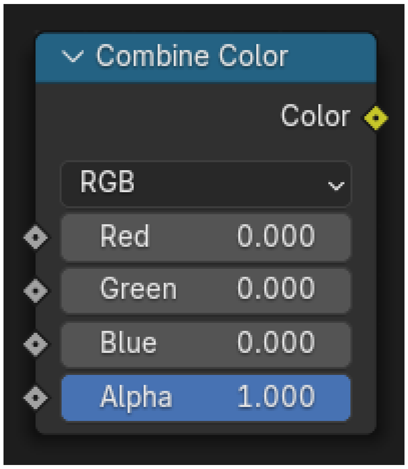

Render Layers
Image
Alpha
Depth
Scene
View Layer
ColorRamp
Image
Alpha
+
−
Fac
Composite
Image

RGB
Linear
2
Pos 1.000
☑ Use Alpha
Alpha 1.000
Z 1.000

The shades of gray in the input image are mapped to three colors: blue, yellow, and red, all fully opaque. Where the image is black, the Color Ramp substitutes blue (the first color stop). Where it is some shade of gray, the Color Ramp outputs a corresponding color from the gradient (bluish, yellow, to reddish). Where the image is fully white, the Color Ramp outputs red.

Combine Color Node

Combines four grayscale channels into one color image, based on a particular Color Model.



RGB

Red 0.000

Green 0.000

Blue 0.000

Alpha 1.000

Inputs

The inputs of this node depend on the Mode property (see below).

Alpha
: The opacity of the output color.

Properties

Mode
: The color model to use.

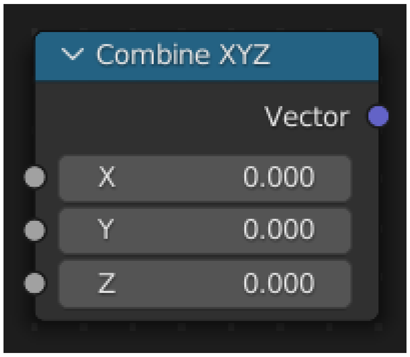
RGB::	Red, Green, Blue.
HSV::	Hue, Saturation, Value.
HSL::	Hue, Saturation, Lightness.

Output

Color
: Standard color output.

Combine XYZ Node

The *Combine XYZ Node* combines a vector from its individual components.



X 0.000
Y 0.000
Z 0.000

Inputs

- X
- Y
- Z

Properties

This node has no properties.

Output

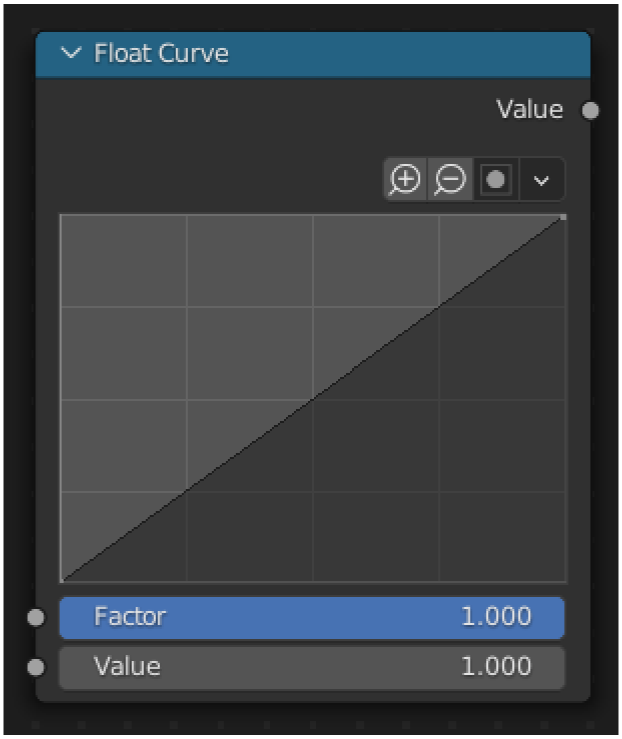
Vector
Standard vector output.

Note

The vector is not normalized.

Float Curve

The Float Curve node maps an input float to a curve and outputs a float value.



Factor 1.000

Value 1.000

Inputs

Factor
: Controls the amount of influence the node exerts on the output value.

Value
: Standard float input.

Properties

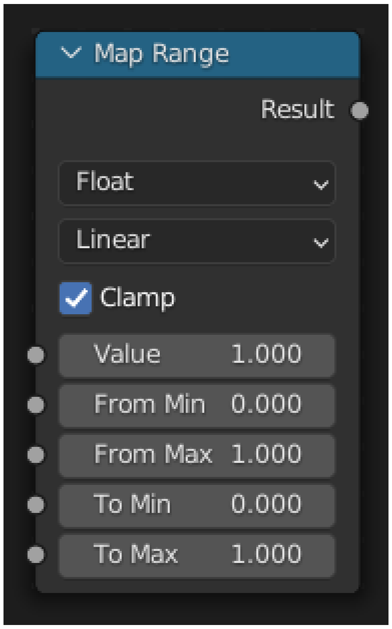
Curve
: For the curve controls see: [Curve widget](#).

Outputs

Float
: Standard float output.

Map Range Node

The *Map Range* node remaps a value from a range to a target range.



Float

Linear

☑ Clamp

Value 1.000

From Min 0.000

From Max 1.000

To Min 0.000

To Max 1.000

Inputs

Value/Vector
: The input value or vector to be remapped.

From Min
: The lower bound of the range to remap from.

From Max
: The higher bound of the range to remap from.

To Min
: The lower bound of the target range.

To Max
: The higher bound of the target range.

Steps

The number of values allowed between *To Min* and *To Max* when using *Stepped Linear* interpolation. A higher value will give a smoother interpolation while lower values will progressively quantize the input.

Properties

Data Type

Map Range supports both Float and Vector data types. Changing the data type will also update the sockets to reflect the data type chosen.

Interpolation Type

The mathematical method used to transition between gaps in the numerical inputs.

Linear::	Linear interpolation between From Min and From Max values.
Stepped Linear::	Stepped linear interpolation between From Min and From Max values.
Smooth Step::	Smooth Hermite edge interpolation between From Min and From Max values.
Smoother Step::	Smoother Hermite edge interpolation between From Min and From Max values.

Clamp

If enabled, the output is clamped to the target range.

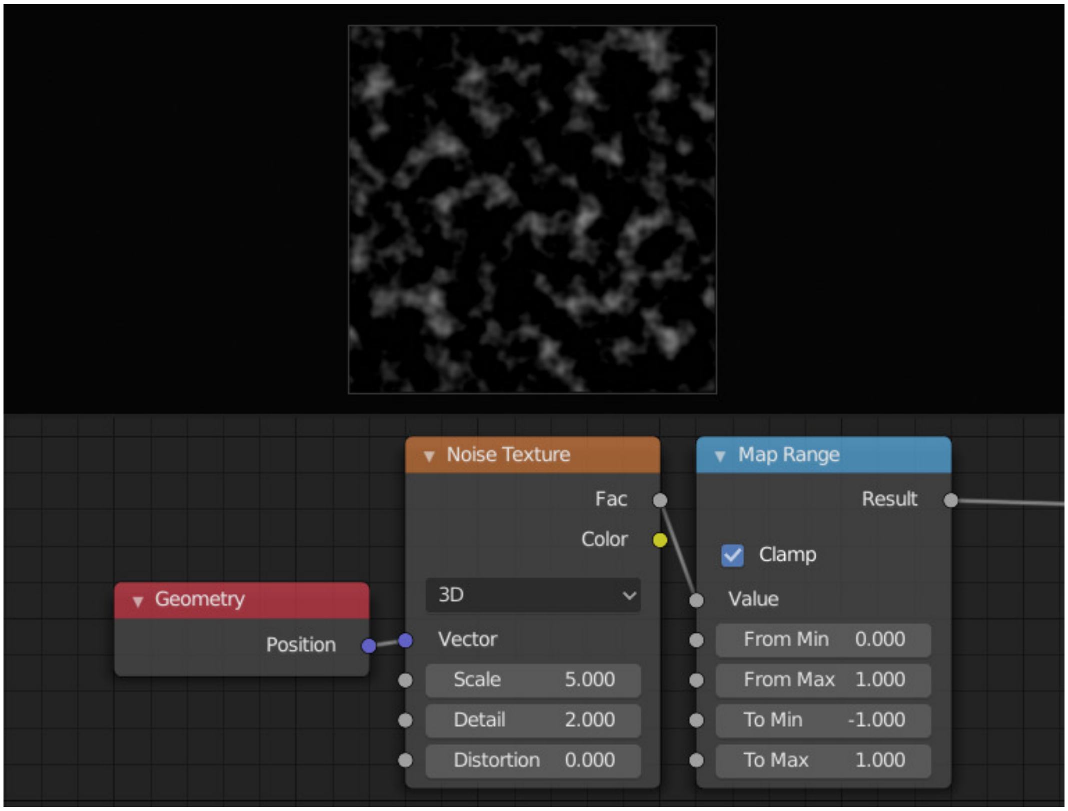
Outputs

Result/Vector

The input value after remapping.

Examples

The *Noise Texture* node outputs a value in the range [0, 1]. We can use the *Map Range* node to remap this value into the range [-1, 1].


Noise Texture
Fac
Color
Map Range
Result
Geometry
Position
Vector
Value

☑ Clamp

3D

Scale 5.000

Detail 2.000

Distortion 0.000

From Min 0.000

From Max 1.000

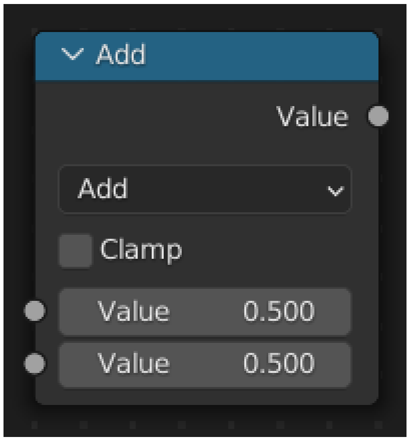
To Min -1.000

To Max 1.000

Example of Map Range node.

Math Node

The *Math Node* performs math operations.



Add

☐ Clamp

Value 0.500

Value 0.500

Inputs

The inputs of the node are dynamic. Some inputs are only available for certain operations. For instance, the *Addend* input is only available for the *Multiply Add* operator.

Value

Input Value. Trigonometric functions read this value as radians.

Addend

Input Addend.

Base

Input Base.

Exponent

Input Exponent.

Epsilon

Input Epsilon.

Distance

Input Distance.

Min

Input Minimum.

Max

Input Maximum.

Increment

Input Increment.

Scale

Input Scale.

Degrees

Input Degrees.

Radians

Input Radians.

Properties

Operation

The mathematical operator to be applied to the input values:

Functions

Add::	The sum of the two values.
Subtract::	The difference between the two values.
Multiply::	The product of the two values.
Divide::	The division of the first value by the second value.
Multiply Add::	The sum of the product of the two values with *Addend*.
Power::	The *Base* raised to the power of *Exponent*.
Logarithm::	The log of the value with a *Base* as its base.
Square Root::	The square root of the value.
Inverse Square Root::	

	One divided by the square root of the value.
Absolute::	The input value is read without regard to its sign. This turns negative values into positive values.
Exponent::	Raises Euler’s number to the power of the value.
Comparison	
Minimum::	Outputs the smallest of the input values.
Maximum::	Outputs the largest of two input values.
Less Than::	Outputs 1.0 if the first value is smaller than the second value. Otherwise the output is 0.0.
Greater Than::	Outputs 1.0 if the first value is larger than the second value. Otherwise the output is 0.0.
Sign::	Extracts the sign of the input value. All positive numbers will output 1.0. All negative numbers will output -1.0. And 0.0 will output 0.0.
Compare::	Outputs 1.0 if the difference between the two input values is less than or equal to *Epsilon*.
Smooth Minimum::	Smooth Minimum.
Smooth Maximum::	Smooth Maximum.
Rounding	
Round::	Rounds the input value to the nearest integer.
Floor::	Rounds the input value down to the nearest integer.
Ceil::	Rounds the input value up to the nearest integer.
Truncate::	Outputs the integer part of the *value*.
Fraction::	Returns the fractional part of the *value*.
Truncated Modulo::	Outputs the remainder once the first value is divided by the second value.
Floored Modulo::	Returns the positive remainder of a division operation.
Wrap::	

	Outputs a value between *Min* and *Max* based on the absolute difference between the input value and the nearest integer multiple of *Max* less than the value.
Snap::	Rounds the input value down to the nearest integer multiple of *Increment*.
Ping-pong::	Bounces back and forth between 0.0 and the *Scale* as the input value increases.

Trigonometric

Sine::	The Sine of the input value.
Cosine::	The Cosine of the input value.
Tangent::	The Tangent of the input value.
Arcsine::	The Arcsine of the input value.
Arccosine::	The Arccosine of the input value.
Arctangent::	The Arctangent of the input value.
Arctan2::	Outputs the Inverse Tangent of the first value divided by the second value measured in radians.
Hyperbolic Sine::	The Hyperbolic Sine of the input value.
Hyperbolic Cosine::	The Hyperbolic Cosine of the input value.
Hyperbolic Tangent::	The Hyperbolic Tangent of the input value.

Conversion

To Radians::	Converts the input from degrees to radians.
To Degrees::	Converts the input from radians to degrees.

Clamp

Limits the output to the range (0.0 to 1.0). See Clamp.

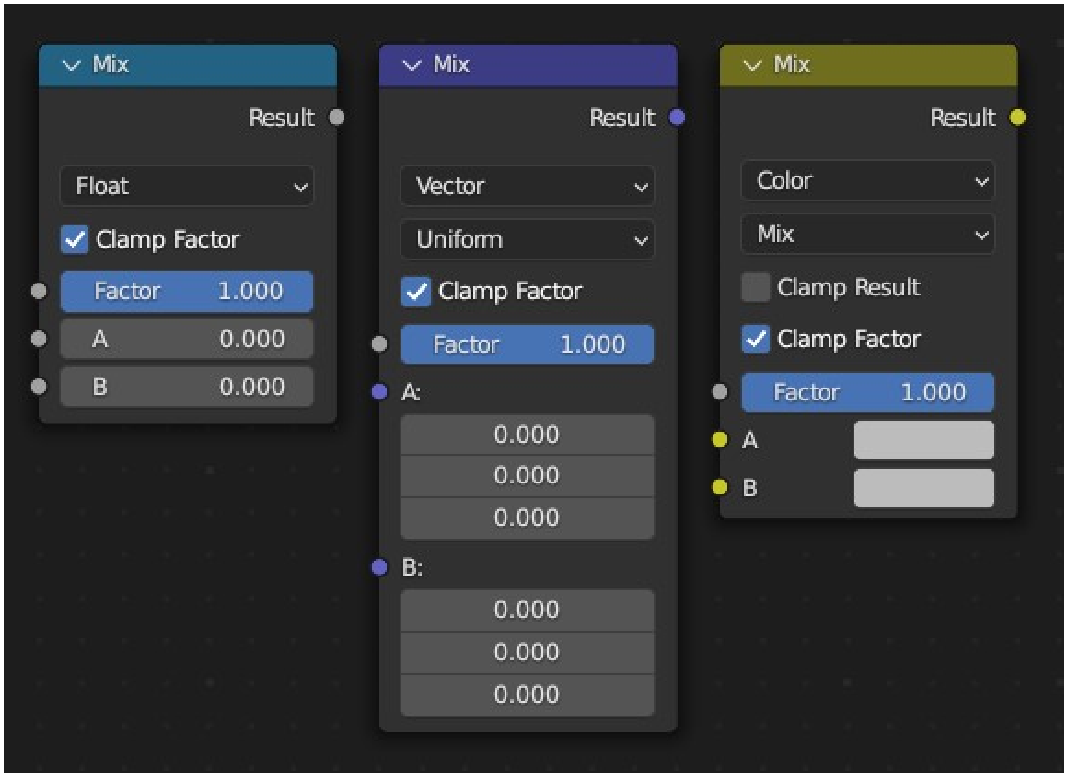
Outputs

Value

Numerical value output.

Mix Node

The *Mix Node* mixes values, colors and vectors inputs using a factor to control the amount of interpolation. The *Color* mode has additional blending modes.


Mix
Result
Float
Clamp Factor
Factor 1.000
A 0.000
B 0.000
Mix
Result
Vector
Uniform
Clamp Factor
Factor 1.000
A:
0.000
0.000
0.000
B:
0.000
0.000
0.000
Mix
Result
Color
Mix
Clamp Result
Clamp Factor
Factor 1.000
A
B

Inputs

Factor
: Controls the amount of mixing between the A and B inputs.

A/B
: The two inputs that are mixed together.

Properties

Data Type

The data type that is used for mixing. The node supports float, vector, color, and rotation data types.

Factor Mode (Vector only)

The factor mode can be set to *Uniform* and *Non-Uniform*. In uniform mode, a single float controls the factor. In non-uniform mode, a vector controls the factor for each XYZ channel separately.

Mix (Color only)

The Blend modes can be selected in the select menu. See Color Blend Modes for details on each blending mode.

Add, Subtract, Multiply, Screen, Divide, Difference, Darken, Lighten, Overlay, Color Dodge, Color Burn, Hue, Saturation, Value, Color, Soft Light, Linear Light

Clamp Factor

Limit the factor value between 0.0 and 1.0. If this option is unchecked then the node operates using *Extrapolation*.

Clamp Result (Color only)

Limit the Result to the range between 0.0 and 1.0.

Outputs

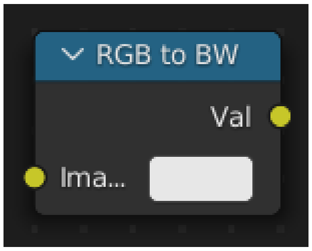
Result

Output the result of the mix using the data type selected.

Examples

See the Color > Mix page for additional examples: Mix Color Node

RGB to BW Node



The *RGB to BW Node* makes a color image black-and-white by outputting its luminance.

Note

You can directly connect Color sockets to Value sockets in node graphs, which also converts the image to black-and-white. As such, this node is not always necessary.

Inputs

Image
    Color image input.

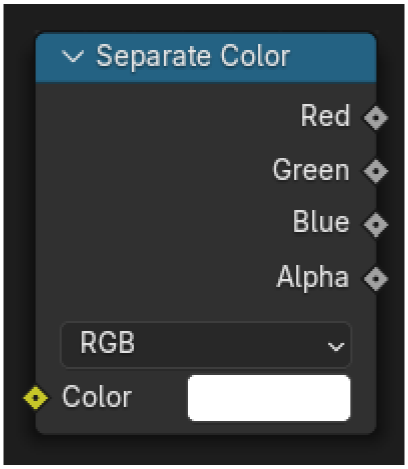
Properties

This node has no properties.

Outputs

Value
    Grayscale value output.

Separate Color Node


Separate Color
Red
Green
Blue
Alpha
RGB
Color

Splits an image into its channels, based on a particular Color Model.

Inputs

Color
: Standard color input.

Properties

Mode
: The color model to output.

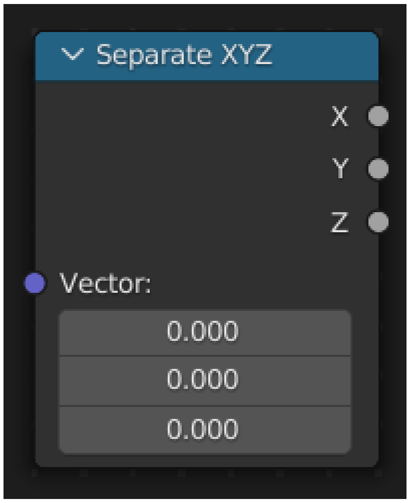
RGB::	Red, Green, Blue.
HSV::	Hue, Saturation, Value.
HSL::	Hue, Saturation, Lightness.

Outputs

The outputs of this node depend on the Mode property (see above).

Alpha
: The opacity value.

Separate XYZ Node


Separate XYZ
X
Y
Z
Vector:
0.000
0.000
0.000

The *Separate XYZ Node* splits a vector into its individual components.

Input

Vector
    Standard vector input.

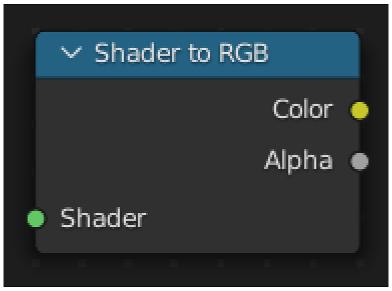
Properties

This node has no properties.

Outputs

- X
- Y
- Z

Shader To RGB



EEVEE Only

The Shader to RGB node is typically used for non-photorealistic rendering, to apply additional effects on the output of BSDFs. For example, a color ramp on the output of a diffuse BSDF can be used to create a flexible toon shader.

Using this conversion breaks the PBR pipeline and thus makes the result unpredictable when used in combination with effects such as ambient occlusion, contact shadows, soft shadows and screen space refraction.

Some effects require multiple samples to converge, and applying arbitrary changes to noisy input may not convert to a smooth result.

Warning

If a Shader to RGB node is used, any upstream BSDF will be invisible to the following effects:

- Screen Space Reflection
- Subsurface Scattering
- Alpha Clip and Alpha Hashed blend modes

Shader to RGB node doesn’t give expected results in render passes.

Inputs

Shader
  Any shader such as a BSDF or Emission node can be linked here.

Properties

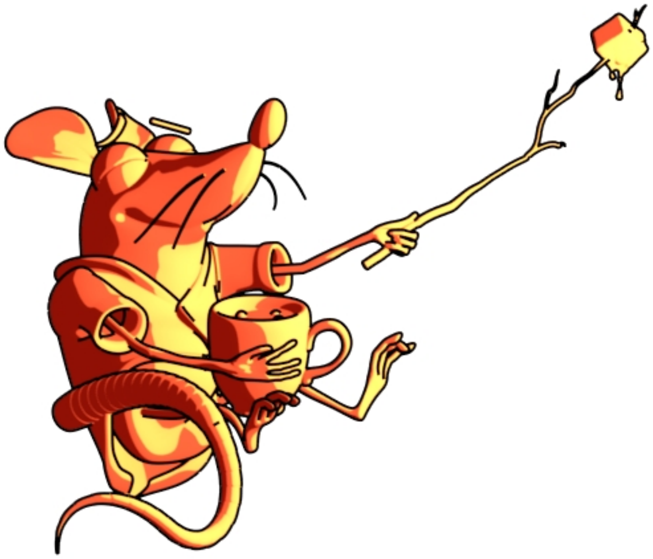
This node has no properties.

Outputs

Color
    Surface color computed from BSDFs and lighting.

Alpha
    Alpha transparency from any Transparent BSDFs in the input.

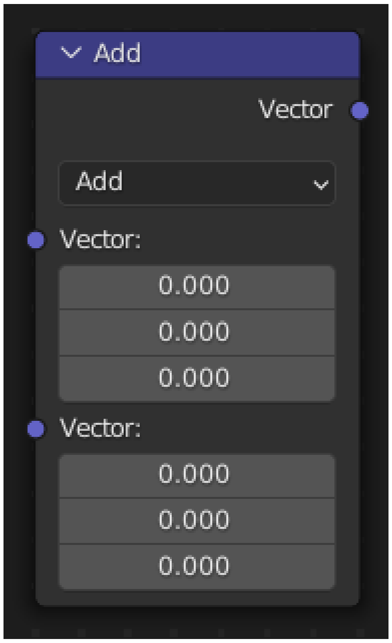
Examples



Simple toon shading with Shader to RGB and Freestyle.

Vector Math Node

The *Vector Math* node performs the selected math operation on the input vectors.


Add
Vector
Add
Vector:
0.000
0.000
0.000
Vector:
0.000
0.000
0.000

Inputs

The inputs of the node are dynamic. Some inputs are only available in certain operations. For instance, the *Scale* input is only available in the *Scale* operator.

Vector
: Input vector $A = \begin{pmatrix} A_x \\ A_y \\ A_z \end{pmatrix}$.

Vector
: Input vector $B = \begin{pmatrix} B_x \\ B_y \\ B_z \end{pmatrix}$.

Scale
: Input Scale s.

Properties

Operation

The vector math operator to be applied on the input vectors.

Add::	The sum of A and B. $\begin{pmatrix} A_x + B_x \\ A_y + B_y \\ A_z + B_z \end{pmatrix}$
Subtract::	The difference between A and B. $\begin{pmatrix} A_x - B_x \\ A_y - B_y \\ A_z - B_z \end{pmatrix}$
Multiply::	The entrywise product of A and B. $\begin{pmatrix} A_x \cdot B_x \\ A_y \cdot B_y \\ A_z \cdot B_z \end{pmatrix}$
Divide::	The entrywise division of A by B. Division by zero results in zero. $\begin{pmatrix} A_x / B_x \\ A_y / B_y \\ A_z / B_z \end{pmatrix}$
Multiply Add::	The entrywise combination of the multiply and addition operations. $A \times B + C$
Cross Product::	The cross product of A and B. $\begin{pmatrix} A_y \cdot B_z - A_z \cdot B_y \\ A_z \cdot B_x - A_x \cdot B_z \\ A_x \cdot B_y - A_y \cdot B_x \end{pmatrix}$
Project::	The projection of A onto B.
Reflect::	The reflection of A around the normal B. B need not be normalized.
Refract::	For a given incident vector A, surface normal B and ratio of indices of refraction (IOR), refract outputs the refraction vector R.
Faceforward::	Orients a vector A to point away from a surface B as defined by its normal C. Computes $(dot(B, C) < 0) ? A : -A$.
Dot Product::	The dot product of A and B. $A_x \cdot B_x + A_y \cdot B_y + A_z \cdot B_z$
Distance::	The distance between A and B.
Length::	The length of A. $\sqrt{A_x^2 + A_y^2 + A_z^2}$
Scale::	The result of multiplying A by the scalar input *Scale*. $\begin{pmatrix} s \cdot A_x \\ s \cdot A_y \\ s \cdot A_z \end{pmatrix}$

Normalize::	The result of normalizing A. The result vector points to the same direction as A and has a length of 1. If A is $(0, 0, 0)$, the result is $(0, 0, 0)$ as well.
Wrap::	The entrywise output of a value between Min and Max based on the absolute difference between the input value and the nearest integer multiple of Max less than the value.
Snap::	The result of rounding A to the largest integer multiple of B less than or equal A.
Floor::	Rounds the input value entrywise down to the nearest integer.
Ceil::	Rounds the input value entrywise up to the nearest integer.
Modulo::	The entrywise modulo of A by B.
Fraction::	Returns the fractional part of the *value* entrywise.
Absolute::	The entrywise absolute value of A.
Minimum::	The entrywise minimum value from A and B.
Maximum::	The entrywise maximum value from A and B.
Sine::	The entrywise Sine of A.
Cosine::	The entrywise Cosine of A.
Tangent::	The entrywise Tangent of A.

Outputs

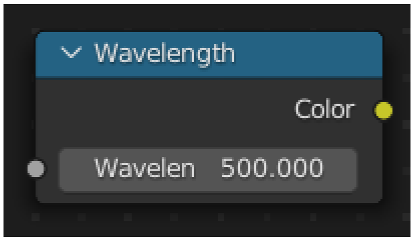
The output of the node is dynamic. It is either a vector or a scalar depending on the operator. For instance, the *Length* operator has a scalar output while the *Add* operator has a vector output.

Vector
    Output vector.

Value
    Output value.

Wavelength Node

The *Wavelength* node converts a wavelength value to an RGB value. This can be used to achieve a specific color on the light spectrum.



Wavelen 500.000

Inputs

Wavelength

The color wavelength from 380 to 780 nanometers.

Properties

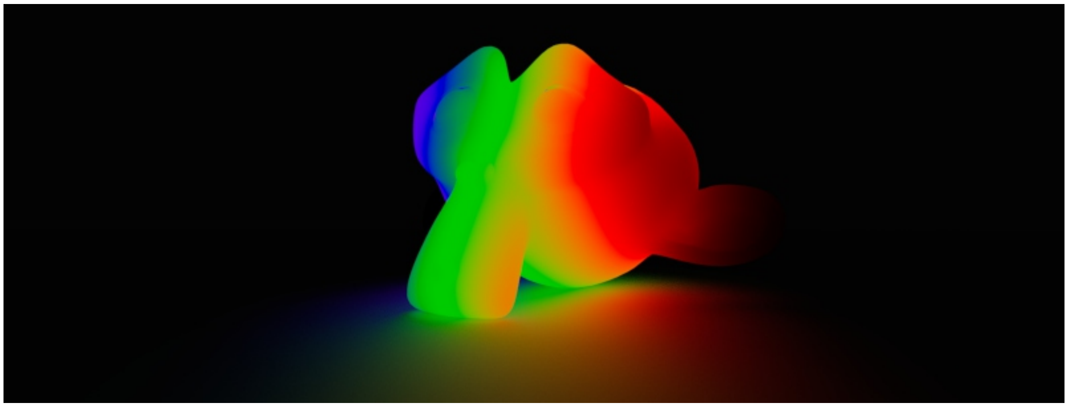
This node has no properties.

Outputs

Color

RGB color output.

Examples



Example of Wavelength node.

Group

A Group Node combines a set of nodes into a single one, and selectively exposes inputs and outputs of those nodes.

Group nodes can simplify a node tree by hiding away complexity and reusing functionality.

Group Input

Exposes the inputs of the node group. You can have multiple of these nodes in your tree to keep it clean, bringing in each input right where you need it (rather than dragging long links all across your graph).

The input slots can be edited in the *Group* tab of the *Sidebar*.

Group Output

Receives the outputs of the node group. You can have multiple of these nodes in your tree to keep it clean, outputting each result right where it's produced (rather than dragging long links all across your graph).

The output slots can be edited in the *Group* tab of the *Sidebar*.

Node Groups

This section lists all the node groups, both those in the current blend-file and those Linked or Appended from another blend-file.

Open Shading Language

Cycles Only

It is also possible to create your own nodes using Open Shading Language (OSL). These nodes will only work with the CPU and OptiX rendering backend.

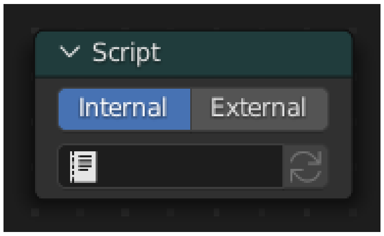
To enable it, select *Open Shading Language* as the shading system in the render settings.

Note

Some OSL features are not available when using the OptiX backend. Examples include:

- Memory usage reductions offered by features like on-demand texture loading and mip-mapping are not available.
- Texture lookups require OSL to be able to determine a constant image file path for each texture call.
- Some noise functions are not available. Examples include *Cell*, *Simplex*, and *Gabor*.
- The trace function is not functional. As a result of this, the Ambient Occlusion and Bevel nodes do not work.

Script Node



OSL was designed for node-based shading, and *each* OSL shader corresponds to *one* node in a node setup. To add an OSL shader, add a script node and link it to a text data-block or an external file. Input and output sockets will be created from the shader parameters on clicking the update button in the Node or the Text editor.

OSL shaders can be linked to the node in a few different ways. With the *Internal* mode, a text data-block is used to store the OSL shader, and the OSO bytecode is stored in the node itself. This is useful for distributing a blend-file with everything packed into it.

The *External* mode can be used to specify a `.osl` file from a drive, and this will then be automatically compiled into a `.oso` file in the same directory. It is also possible to specify a path to a `.oso` file, which will then be used directly, with compilation done manually by the user. The third option is to specify just the module name, which will be looked up in the shader search path.

The shader search path is located in the same place as the scripts or configuration path, under:

Linux

```
$HOME/.config/blender/4.3/shaders/
```

Windows

```
C:\Users\$user\AppData\Roaming\Blender
Foundation\Blender\4.3\shaders\
```

macOS

```
/Users/$USER/Library/Application Support/Blender/4.3/shaders/
```

Tip

For use in production, we suggest to use a node group to wrap shader script nodes, and link that into other blend-files. This makes it easier to make changes to the node afterwards as sockets are added or removed, without having to update the script nodes in all files.

Writing Shaders

For more details on how to write shaders, see the OSL Documentation.

Here is a simple example:

```
shader simple_material(
    color Diffuse_Color = color(0.6, 0.8, 0.6),
    float Noise_Factor = 0.5,
    output closure color BSDF = diffuse(N))
{
    color material_color = Diffuse_Color * mix(1.0, noise(P *
10.0), Noise_Factor);
    BSDF = material_color * diffuse(N);
}
```

Closures

OSL is different from, for example, RSL or GLSL, in that it does not have a light loop. There is no access to lights in the scene, and the material must be built from closures that are implemented in the renderer itself. This is more limited, but also makes it possible for the renderer to do optimizations and ensure all shaders can be importance sampled.

The available closures in Cycles correspond to the shader nodes and their sockets; for more details on what they do and the meaning of the parameters, see the shader nodes manual.

See also

Documentation on OSL's built-in closures.

BSDF

- `diffuse(N)`
- `oren_nayar(N, roughness)`
- `diffuse_ramp(N, colors[8])`
- `phong_ramp(N, exponent, colors[8])`
- `diffuse_toon(N, size, smooth)`
- `glossy_toon(N, size, smooth)`
- `translucent(N)`
- `reflection(N)`
- `refraction(N, ior)`

- `transparent()`
- `microfacet_ggx(N, roughness)`
- `microfacet_ggx_aniso(N, T, ax, ay)`
- `microfacet_ggx_refraction(N, roughness, ior)`
- `microfacet_beckmann(N, roughness)`
- `microfacet_beckmann_aniso(N, T, ax, ay)`
- `microfacet_beckmann_refraction(N, roughness, ior)`
- `ashikhmin_shirley(N, T, ax, ay)`
- `ashikhmin_velvet(N, roughness)`

Hair

- `hair_reflection(N, roughnessu, roughnessv, T, offset)`
- `hair_transmission(N, roughnessu, roughnessv, T, offset)`
- `principled_hair(N, absorption, roughness, radial_roughness, coat, offset, IOR)`

BSSRDF

Used to simulate subsurface scattering.

$\mathrm{bssrdf}(method, N, radius, albedo)$

Parameters::

- **method** (*string*) –
  Rendering method to simulate subsurface scattering.
  - `burley`: An approximation to physically-based volume scattering. This method is less accurate than `random_walk` however, in some situations this method will resolve noise faster.
  - `random_walk_skin`: Provides accurate results fc thin and curved objects. Random Walk uses true volumetric scattering inside the mesh, which means that it works best for closed meshes. Overlapping faces and holes in the mesh can cause problems.
  - `random_walk`: Behaves similarly to `random_walk_skin` but modulates the *Radius* based on the *Color*, *Anisotropy*, and *IOR*. This method thereby attempts to retain greater surfac detail and color than `random_walk_skin`.
- **N** (*vector*) – Normal vector of the surface point being shaded.
- **radius** (*vector*) – Average distance that light scatters below the surface. Higher radius gives a softer appearance, as light bleeds into shadows an through the object. The scattering distance is specified separately for the RGB channels, to render materials such as skin where red light scatters deeper. The X, Y and Z values are mappec to the R, G and B values, respectively.
- **albedo** (*color*) – Color of the surface, or physicall speaking, the probability that light is reflected for each wavelength.

Volume

- `henyey_greenstein(g)`
- `absorption()`

Other

- `emission()`
- `ambient_occlusion()`
- `holdout()`
- `background()`

Attributes

Geometry attributes can be read through the `getattribute()` function. This includes UV maps, color attributes and any attributes output from geometry nodes.

The following built-in attributes are available through `getattribute()` as well.

`geom:generated`	Automatically generated texture coordinates, from undeformed mesh.
`geom:uv`	Default render UV map.
`geom:tangent`	Default tangent vector along surface, in object space.
`geom:undisplaced`	Position before displacement, in object space.
`geom:dupli_generated`	For instances, generated coordinate from instancer object.
`geom:dupli_uv`	For instances, UV coordinate from instancer object.
`geom:trianglevertices`	Three vertex coordinates of the triangle.
`geom:numpolyvertices`	Number of vertices in the polygon (always returns three currently).
`geom:polyvertices`	

Vertex coordinates array of the polygon (always three vertices currently).

`geom:name`
Name of the object.

`geom:is_smooth`
Is mesh face smooth or flat shaded.

`geom:is_curve`
Is object a curve or not.

`geom:curve_intercept`
0..1 coordinate for point along the curve, from root to tip.

`geom:curve_thickness`
Thickness of the curve in object space.

`geom:curve_length`
Length of the curve in object space.

`geom:curve_tangent_normal`
Tangent Normal of the strand.

`geom:is_point`
Is point in a point cloud or not.

`geom:point_radius`
Radius of point in point cloud.

`geom:point_position`
Center position of point in point cloud.

`geom:point_random`
Random number, different for every point in point cloud.

`path:ray_length`
Ray distance since last hit.

`object:random`
Random number, different for every object instance.

`object:index`
Object unique instance index.

`object:location`
Object location.

`material:index`
Material unique index number.

`particle:index`
Particle unique instance number.

`particle:age`
Particle age in frames.

`particle:lifetime`
Total lifespan of particle in frames.

`particle:location`
Location of the particle.

`particle:size`
Size of the particle.

`particle:velocity`
Velocity of the particle.

`particle:angular_velocity`
Angular velocity of the particle.

Trace

CPU Only

We support the `trace(point pos, vector dir, ...)` function, to trace rays from the OSL shader. The "shade" parameter is not supported currently, but attributes can be retrieved from the object that was hit using the `getmessage("trace", ..)` function. See the OSL specification for details on how to use this.

This function cannot be used instead of lighting; the main purpose is to allow shaders to “probe” nearby geometry, for example to apply a projected texture that can be blocked by geometry, apply more “wear” to exposed geometry, or make other ambient occlusion-like effects.

Metadata

Metadata on parameters controls their display in the user interface. The following metadata is supported:

`[[ string label = "My Label" ]]`
  Name of parameter in in the user interface

`[[ string widget = "null" ]]`
  Hide parameter in the user interface.

`[[ string widget = "boolean" ]]` and `[[ string widget = "checkbox" ]]`
  Display integer parameter as a boolean checkbox.

Color Management

Color management is important to create renders and assets that are physically accurate and look great on multiple display devices. It is used both to ensure all parts of the pipeline interpret colors correctly, and to make artistic changes like exposure and color grading.

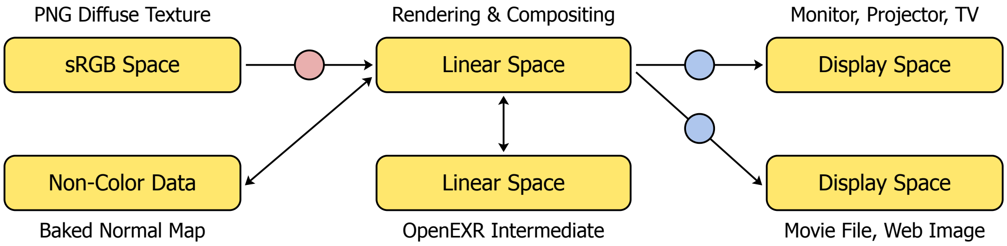
Blender's color management is based on the OpenColorIO library. By using the same OpenColorIO configuration in multiple applications, the same color spaces and transforms will be available for consistent results.

Different views and exposures of the same render.

Workflow

Scene Linear Color Space

For correct results, different Color Spaces are needed for rendering, display and storage of images. Rendering and compositing is best done in *scene linear* color space, which corresponds more closely to nature, and makes computations more physically accurate.


PNG Diffuse Texture
sRGB Space
Non-Color Data
Baked Normal Map
Rendering & Compositing
Linear Space
Linear Space
OpenEXR Intermediate
Monitor, Projector, TV
Display Space
Display Space
Movie File, Web Image

An example of a linear workflow.

If the colors are linear, it means that if in reality, we double the number of photons, the color values are also doubled. Put another way, if we have two photos/renders each with one of two lights on, and add those images together, the result would be the same as a render/photo with both lights on. It follows that such a radiometrically linear space is best for photorealistic rendering and compositing.

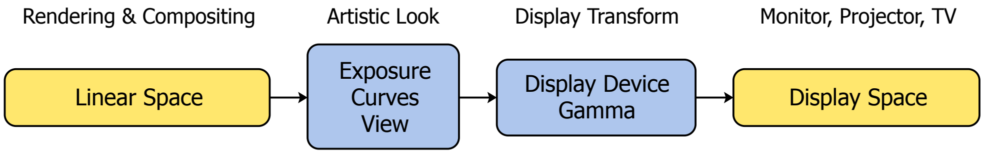
However, these values do not directly correspond to human perception or the way display devices work. and image files are often stored in different color spaces. So we have to take care to do the right conversion into and out of this scene linear color space.

Display Transforms

Transforming scene linear colors to display involves both technical and artistic choices.

Correct display of renders requires a conversion to the display device color space. A computer monitor works differently from a digital cinema projector or HDTV, and so needs a different conversion.

There is also an artistic choice to be made. Partially that is because display devices cannot display the full spectrum of colors and only have limited brightness, so we can squeeze the colors to fit in the gamut of the device. Besides that, it can also be useful to give the renders a particular look, e.g. as if they have been printed on real camera film. The default Filmic transform does this.


Rendering & Compositing
Artistic Look
Display Transform
Monitor, Projector, TV
Linear Space
Exposure
Curves
View
Display Device
Gamma
Display Space

Conversion from linear to display device space.

Image Color Spaces

When loading and saving media formats it is important to have color management in mind. File formats such as PNG or JPEG will typically store colors in a color space ready for display, not in a linear space. When they are used as textures in renders, they need to be converted to linear first, and when saving renders for display on the web, they also need to be converted to a display space.

For intermediate files in production, it is recommended to use *OpenEXR* files. These are always stored in scene linear color spaces, without any data loss. That makes them suitable to store renders that can later be composited, color graded and converted to different output formats.

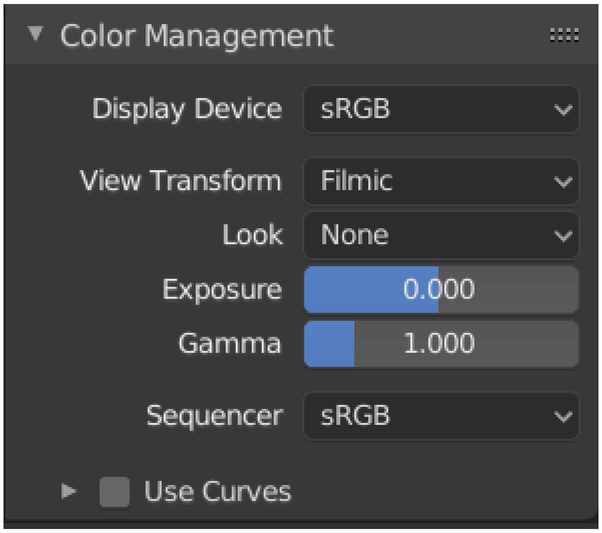
Images can also contain data that is not actually a color. For example normal or displacement maps merely contain vectors and offsets. Such images should be marked as *Non-Color Data* so that no color space conversion happens on them.

Render Settings

Reference

Editor::	Properties
Panel::	*Render Properties* ▸ *Color Management*

These are color management settings that are used across Blender. These color management settings are Scene specific so settings can be customized



Display Device sRGB
View Transform Filmic
Look None
Exposure 0.000
Gamma 1.000
Sequencer sRGB
☐ Use Curves

Color Management properties.

per Scene. Color management can also be overridden when saving images; this behavior can be set in the [Output Color Management Properties](#).

Display Device

The color space for the display that Blender is being viewed on.

Most displays are sRGB by default with some newer displays having the option to use Rec. 2020. These displays have a wider color gamut and can display high dynamic range content. If you have an Apple display you probably will want to use Display P3.

It is important to check your OS and display setting to make sure they all match the display in use to view the most accurate image.

sRGB::	Used by most displays.
Display P3::	Used by most Apple devices.
Rec. 1886::	Used by many older TVs.
Rec. 2020::	Used for newer wide gamut HDR displays.

View Transform

These are different ways to view the image on the same display device.

Standard::	Does no extra conversion besides the conversion for the display device. Often used for non-photorealistic results or video editing where a specific look is already baked into the input video.
Khronos PBR Neutral::	

A tone mapping transform designed specifically for PBR color accuracy, to get sRGB colors in the output render that match as faithfully as possible the input sRGB base color in materials, under gray-scale lighting. This is aimed toward product photography use cases, where the scene is well-exposed and HDR color values are mostly restricted to small specular highlights.

AgX:: A tone mapping transform that improves on *Filmic*, giving more photorealistic results. AgX offers 16.5 stops of dynamic range and unsaturates highly exposed colors to mimic film's natural response to light.

Filmic:: A tone mapping transform designed to handle high dynamic range colors. Filmic is deprecated and is superseded by AgX which improves handling of saturated colors.

Filmic Log:: Converts to Filmic log color space. This can be used for export to color grading applications, or to inspect the image by flattening out very dark and light areas.

False Color:: Shows a heat map of image intensities, to visualize the dynamic range, and help properly expose an image.

Below is a table that represents how normalized linear color data is represented with False Color.

Luminance Value	Color
Low Clip	Black
0.0001% to 0.05%	Blue
0.05% to 0.5%	Blue-Cyan
0.5% to 5%	Cyan
5% to 16%	Green-Cyan
16% to 22%	Grey
22% to 35%	Green-Yellow

Luminance Value	Color
35% to 55%	Yellow
55% to 80%	Orange
80% to 97%	Red
High Clip	White

Raw:: Intended for inspecting the image but not for final export. Raw gives the image without any color space conversion.

Look

Choose an artistic effect from a set of measured film response data which roughly emulates the look of certain film types. Applied before color space conversion.

Exposure

Used to control the image brightness (in stops) applied before color space conversion. It is calculated as follows: \(output_value = render_value × 2^{(exposure)}\)

Gamma

Extra gamma correction applied after color space conversion. Note that the default display transforms already perform the appropriate conversion, so this mainly acts as an additional effect for artistic tweaks.

Sequencer

The color space that the Sequencer operates in. By default, the Sequencer operates in sRGB space, but it can also be set to work in Linear space like the Compositing nodes, or another color space. Different color spaces will give different results for color correction, crossfades, and other operations.

The list of color spaces depends on the active OCIO config. The default supported color spaces are described in detail here: Default OpenColorIO Configuration

Display

High Dynamic Range

Enable high dynamic range display in rendered viewport, uncapping display brightness. This requires a monitor with HDR support and a view transform designed for HDR (Filmic does not generate HDR colors).

This feature is currently only supported on macOS.

Use Curves

Adjust RGB Curves to control the image colors before the color space conversion. Read more about using the Curve Widget.

White Balance

Adjusts colors so that a given white point (expressed in color temperature and tint) ends up as white on the display.

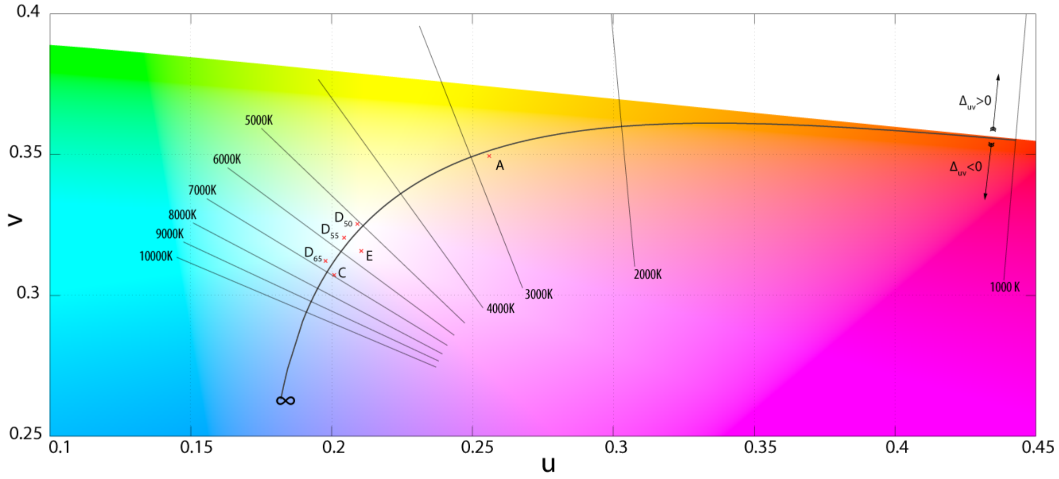
As an alternative to manually specifying the values, there's also a color picker. When a color is selected, temperature and tint are set such that this color ends up being balanced to white. This only works if the color is close enough to a blackbody emitter.

Temperature

The blackbody temperature of the primary illuminant. By default a D65 white point is used.

Tint

The amount of green/magenta shift of the blackbody curve.



Blackbody temperature curve.

Tip

White balancing can also be accomplished as part of the compositing pipeline by using the Color Balance Node

Image Files

When working with image files, the default color space is usually the right one. If this is not the case, the color space of the image file can be configured in the image settings. A common situation where manual changes are needed is when working with or baking normal maps or displacement maps, for example. Such maps do not actually store colors, just data encoded as colors. Those images should be marked as *Non-Color Data*.

Image data-blocks will always store float buffers in memory in the scene linear color space, while a byte buffer in memory and files in a drive are stored in the specified color space setting.

By default, only renders are displayed and saved with the render *View Transformation* applied. These images are the “Render Result” and “Viewer” image data-blocks, and the files saved directly to a drive with the

Render Animation operator. However, when loading a render saved to an intermediate OpenEXR file, Blender cannot detect automatically that this is a render (it could be e.g. an image texture or displacement map). We need to specify that this is a render and that we want the transformations applied, with these two settings:

View as Render
: Display the image data-block (not only renders) with view transform, exposure, gamma, RGB curves applied. Useful for viewing rendered frames in linear OpenEXR files the same as when rendering them directly.

Save as Render
: Option in the image save operator to apply the view transform, exposure, gamma, RGB curves. This is useful for saving linear OpenEXR to e.g. PNG or JPEG files in display space.

OpenColorIO Configuration

Blender comes with a standard OpenColorIO configuration that contains a number of useful display devices and view transforms. The reference linear Color Space used is the linear color space with Rec. 709 chromaticities and D65 white point.

However, OpenColorIO is also designed to give a consistent user experience across multiple applications, and for this, a single shared configuration file can be used. Blender will use the standard OCIO environment variable to read an OpenColorIO configuration other than the default Blender one. More information about how to set up such a workflow can be found on the OpenColorIO website.

Blender currently use the following color space rules:

`scene_linear`
: Color space used for rendering, compositing, and storing all float precision images in memory.

`data`
: Color space for non-color data.

`aces_interchange`
: ACES2065-1 color space. Used to derive chromaticities of the *scene_linear* color space, for effects such as blackbody emission.

`color_picking`
: Defines the distribution of colors in color pickers. It is expected to be approximately perceptually linear, have the same gamut as the *scene_linear* color space, map 0..1 values to 0..1 values in the scene linear color space for predictable editing of materials' albedo.

`default_sequencer`
: Default color space for the Sequencer, *scene_linear* if not specified.

`default_byte`
: Default color space for byte precision images and files, *texture_paint* if not specified.

`default_float`
: Default color space for float precision images and files, *scene_linear* if not specified.

The standard Blender configuration includes support for saving and loading images in ACES (code and documentation) color spaces. However, the ACES gamut is larger than the Rec. 709 gamut, so for best results, an ACES specific configuration file should be used. OpenColorIO provides an ACES configuration file, though it may need a few more tweaks to be usable in production.

Default OpenColorIO Configuration

Color Spaces
: Blender's OCIO configuration file is equipped by default to read/write files in these color spaces:

sRGB::	Standard RGB display space using Rec. 709 chromaticities and a D65 white point.
Rec.2020::	BT.2020 2.4 Exponent EOTF Display.
Rec.1886::	BT.1886 2.4 Exponent EOTF Display, commonly used for TVs.
Non-Color::	Generic data that is not color, will not apply any color transform (e.g. normal maps).
Linear Rec.709::	Linear BT.709 chromaticities with illuminant D65 white point.
Linear Rec.2020::	Linear BT.2020 with illuminant D65 white point.
Linear FilmLight E-Gamut::	Linear E-Gamut with illuminant D65 white point.
Linear DCI-P3 D65::	Linear DCI-P3 with illuminant D65 white point.
Linear CIE-XYZ E::	1931 CIE XYZ standard with assumed illuminant E white point.
Linear CIE-XYZ D65::	1931 CIE XYZ with adapted illuminant D65 white point.
Filmic sRGB::	Similar to *sRGB* but uses the Filmic view transform.
Filmic Log::	Intermediate log color space of Filmic view transform.
Display P3::	Apple’s Display P3 with sRGB compound (piece-wise) encoding transfer function, common on Mac devices.
ACEScg::	

	An ACES color space that is designed to be used for rendering and compositing. It uses the AP1 color primaries, a D60 white point, and a linear transfer function. While similar to ACES2065-1, this color space has a smaller color gamut. The smaller gamut allow it to better represent the colors that fit inside the CIE 1931 chromaticities diagram. Colors that lie outside the CIE 1931 chromaticities are generally not important to rendering and compositing because the human stimulus cannot represent these colors.
ACES2065-1::	An ACES color space using the AP0 color primaries, a D60 white point and a linear transfer function. This color space is meant to store and transfer data with the most amount of possible color information.

Freestyle

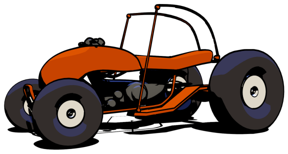
- Introduction
  - The Big Picture
  - Known Limitations
- Render Properties
- View Layer Properties
  - Freestyle
  - Line Set
  - Line Style
- Material Properties
- Python Scripting
  - Writing Style Modules

Introduction

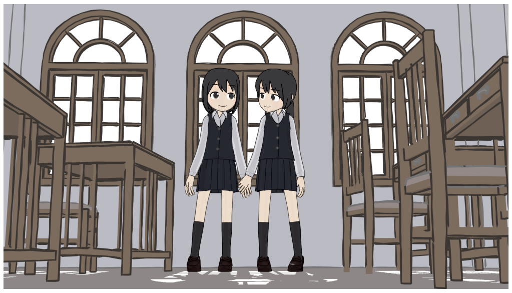
Freestyle is an edge/line-based non-photorealistic (NPR) rendering engine. It relies on mesh data and Z-depth information to draw lines on selected edge types. Various line styles can be added to produce artistic ("hand drawn", "painted", etc.) or technical (hard line) looks.

Freestyle can generate a powerful diversity of line styles and results. There are currently, two ways to define the way lines look; the first uses a series of parameter to create a Line Style. This mode allows intuitive editing of features such as dotted lines and easy setup of multiple line types and edge definitions. On top of all of that, with line style modifiers, the sky is the limit!

The second method of generating lines is by using Python Scripting. This method is much more advanced but Blender includes many pre-scripted styles such as Japanese big brush, cartoon, blueprint, and thickness-with-depth.

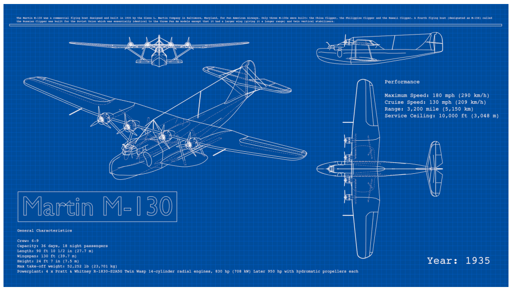


ATV buggy by Rylan Wright (RONIN). CC BY. (File:AtvBuggy.zip)



By mato.sus304. CC BY-SA. (File:Mato_sus304_cut02.zip)

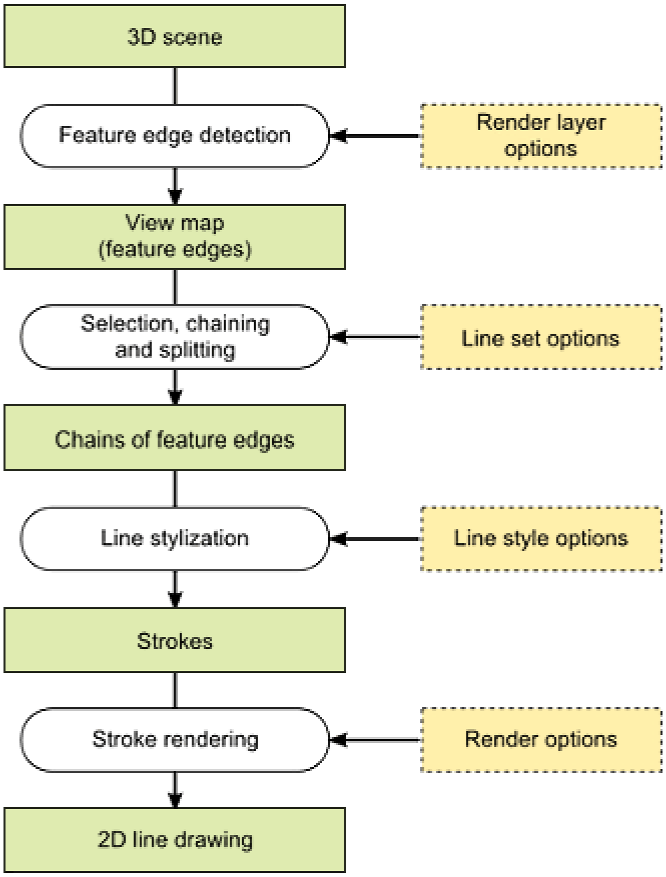
A cartoon scene from OHA Studio © Mechanimotion Entertainment. (blend-file)


Performance
Maximum Speed: 180 mph (290 km/h)
Cruise Speed: 130 mph (209 km/h)
Range: 3,200 mile (5,150 km)
Service Ceiling: 10,000 ft (3,048 m)
Martin M-130
General Characteristics
Crew: 6-9
Capacity: 36 days, 18 night passengers
Length: 90 ft 10 1/2 in (27.7 m)
Wingspan: 130 ft (39.7 m)
Height: 24 ft 7 in (7.5 m)
Max take-off weight: 52,252 lb (23,701 kg)
Powerplant: 4 x Pratt & Whitney R-1830-S2A5G Twin Wasp 14-cylinder radial engines, 830 hp (708 kW) Later 950 hp with hydromatic propellers each
Year: 1935

Blueprint render of Martin M-130 from 1935 by LightBWK. CC0. Warning: heavy file! designed for stress test Blender to the limits and may crash Blender. (File:M-130Blueprint.zip)

The Big Picture

- Activate Freestyle by the *Properties* ▸ *Render* ▸ *Freestyle* checkbox.
- Freestyle settings are located in the View Layer properties.
- One view layer can only have one view map. A view map holds the edge detection settings (Crease Angle, Culling toggle, Face Smoothness toggle, Material Boundaries toggle, Sphere Radius, and Kr Derivative Epsilon advanced options).
- A view map can have multiple Line Sets.
- A line set controls which line types and selections will be rendered, from lines based on your scene.
- Each line set uses one line style (which can be shared between multiple Line Sets).
- A line style tells Freestyle how to render the linked Line Sets in terms of color, alpha, thickness and other aspects.


3D scene
Feature edge detection
Render layer options
View map (feature edges)
Selection, chaining and splitting
Line set options
Chains of feature edges
Line stylization
Line style options
Strokes
Stroke rendering
Render options
2D line drawing

Block diagram of Freestyle view map and processes.

Known Limitations

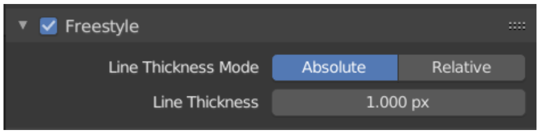
- Highly memory demanding: All mesh objects in a view layer are loaded at once.
- Only faced mesh objects are supported.
- No edges at face intersections are detected yet.
- Freestyle rendering results do not have any Z depth information.
- Panoramic cameras are not supported.

Render Properties

Reference

Panel:: *Properties* ▸ *Render* ▸ *Freestyle*

Freestyle can be activated with the checkbox in the header of the Freestyle panel in the *Render* tab.



☑ Freestyle

Line Thickness Mode: [✓ Absolute] Relative

Line Thickness: 1.000 px

Freestyle Render Properties.

Line Thickness Mode

There are two different modes for defining the base line thickness:

Absolute::	The line thickness is given by a user-specified number of pixels.
Relative::	The unit line thickness is scaled by the proportion of the present vertical image resolution to 480 pixels. For instance, the “unit line thickness” is 1.0 when the image height set to 480px, 1.5 with 720px and 2.0 with 960px.

Line Thickness

Line thickness to use for rendering (only for *Absolute* line thickness).

View Layer Properties

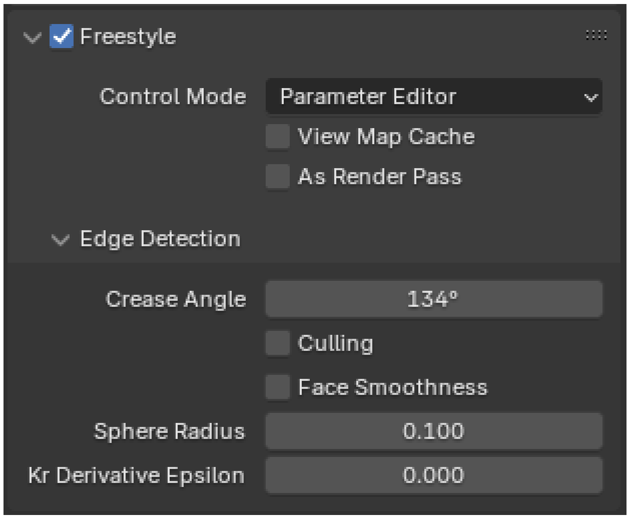
- Freestyle
  - Edge Detection
- Line Set
  - Visibility
  - Edge Types
  - Face Marks
  - Collection
- Line Style
  - Introduction
  - Properties
  - Modifiers

Freestyle

Reference

Panel:: *Properties ‣ View Layer ‣ Freestyle*

There is only one view map per view layer. It controls the edge detection parameters. Freestyle can be enabled/disabled per View Layer by toggling the checkbox in the panel header.



☑ Freestyle

Control Mode: Parameter Editor

☐ View Map Cache

☐ As Render Pass

Edge Detection

Crease Angle: 134°

☐ Culling

☐ Face Smoothness

Sphere Radius: 0.100

Kr Derivative Epsilon: 0.000

View Layer: Freestyle panel.

Control Mode

Which detected edges are actually rendered, and how, can be controlled either through:

Parameter Editor Mode::

Lines are rendered via parameters defined in a user-friendly interface to define and control Line Sets and line styles.

A view map (hence a view layer) can have multiple Line Sets, and each line set is linked to one line style.

Python Scripting Mode::

Lines are rendered via Python scripting, powerful but complex.

View Map Cache

An option to reuse a previously computed view map for subsequent rendering. The cache is automatically updated when the mesh geometry of the input 3D scene has been changed.

This functionality offers a major performance boost for Freestyle animation rendering when the camera-space mesh geometry is static, as well as for repeated still renders with updates of line stylization options.

Although the *View Map Cache* checkbox is a view layer option, the cache memory is shared by all view layers and scenes. This means that if Freestyle is used for two or more view layers (possibly in different scenes through the Compositor), then the cached view map for one view layer is replaced by a new view map for another view layer and hence no performance gain is expected.

As Render Pass

Freestyle lines will not immediately be visible on top of the render image. Instead, Freestyle lines are rendered as a Render Pass which can be composited with the rendered image with an Alpha Over node.

Edge Detection

Crease Angle

If two adjacent faces form an angle less than the defined *Crease Angle*, the edge between them will be rendered when using *Crease* edge type selection in a line set. The value also affects *Silhouette* edge type selection.

Culling

Ignore the edges that are out of view. (Saves some processing time and memory, but may reduce the quality of the result in some cases.)

Face Smoothness

Takes *Smooth Shading* into account for edges calculation.

Sphere Radius

Affects the calculation of curvatures for *Ridge*, *Valley* and *Suggestive Contour* edge type selection in a line set. The curvature at each vertex is computed by averaging the shape of the surface within the specified radius. Increasing the value reduces noise and detail.

Kr Derivative Epsilon

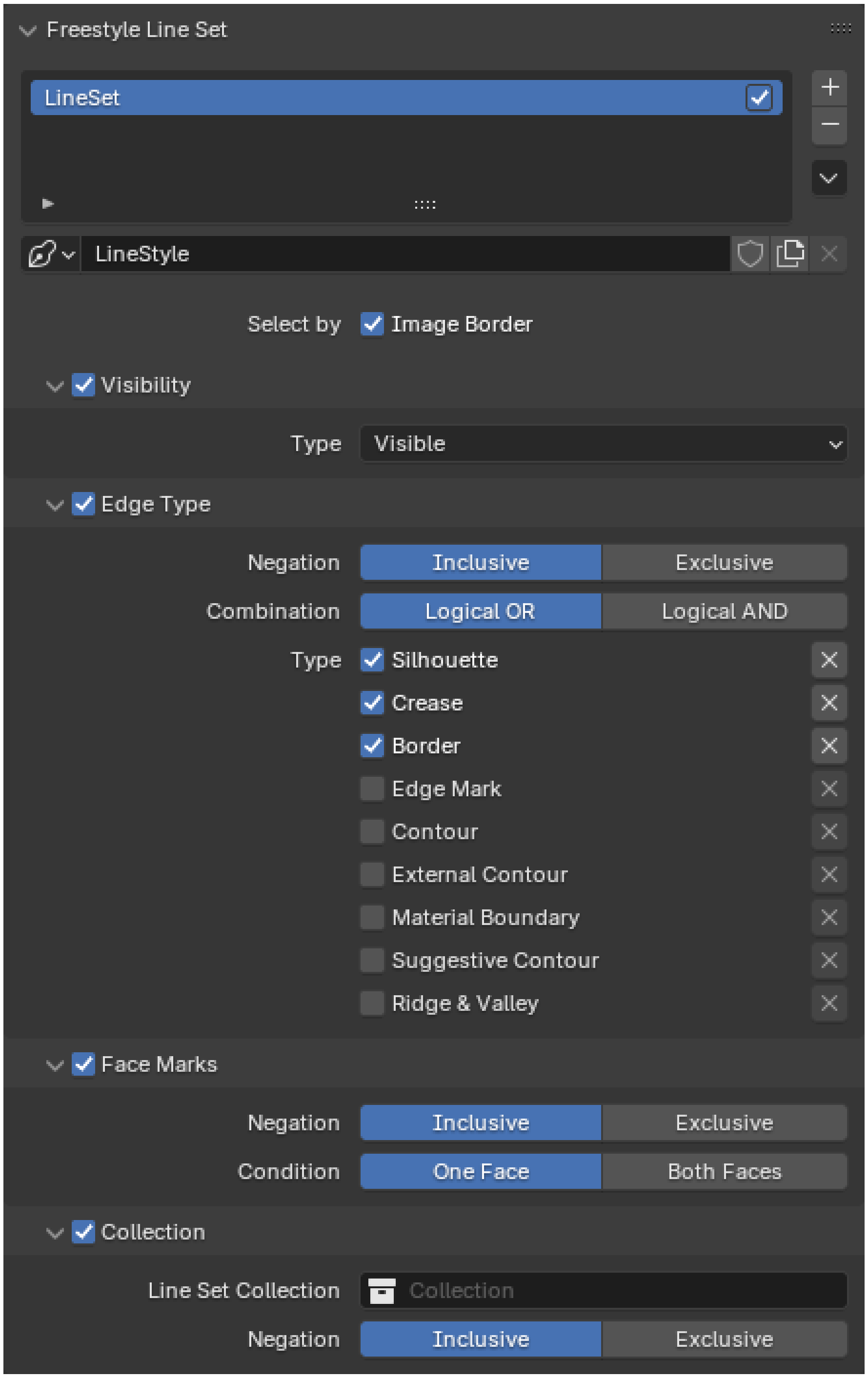
Controls the threshold on the minimum rate of change of curvature used to filter the output of the *Suggestive Contour* edge type selection. Increasing the value reduces the amount of rendered lines, starting from smoother areas of the object (further information in this pdf).

Line Set

Reference

Panel:: *Properties* ▸ *View Layer* ▸ *Freestyle Line Set*

A line set selects, among the lines (edges) detected by Freestyle, which ones will be rendered using its attached line style, through various methods.



Freestyle Line Set

LineSet ☑

LineStyle

Select by ☑ Image Border

☑ Visibility

Type: Visible

☑ Edge Type

Negation: **Inclusive** | Exclusive

Combination: **Logical OR** | Logical AND

Type:
- ☑ Silhouette
- ☑ Crease
- ☑ Border
- ☐ Edge Mark
- ☐ Contour
- ☐ External Contour
- ☐ Material Boundary
- ☐ Suggestive Contour
- ☐ Ridge & Valley

☑ Face Marks

Negation: **Inclusive** | Exclusive

Condition: **One Face** | Both Faces

☑ Collection

Line Set Collection: Collection

Negation: **Inclusive** | Exclusive

Freestyle Line Set panel.

Select By

Image Border

Causes Freestyle to only take geometry within the image border into consideration for line calculation. This reduces render times but increases continuity problems when geometry is moved out of and into camera view.

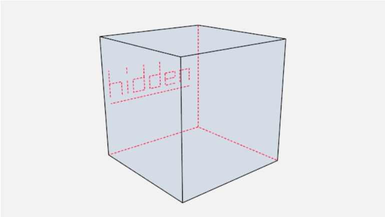
Visibility

Type

Determine how to use visibility for feature edge selection.

Visible:: Only lines occluded by no surfaces are rendered.

Hidden:: Lines occluded by at least one surface are rendered.



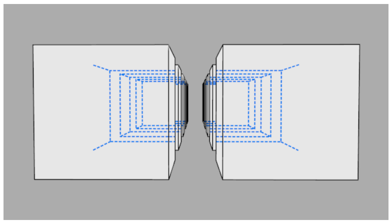
Proof of concept of visible and hidden edges by LightBWK (blend-file).

Quantitative Invisibility::

Lines occluded by a number of surfaces in the given range are rendered.

Start, End

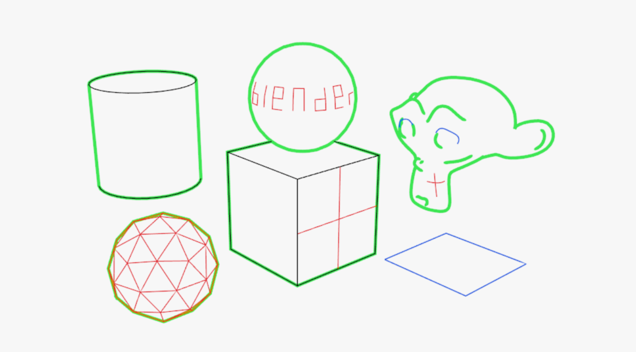
Min/max number of occluding surfaces for a line to be rendered.



QI Range proof of concept demo, Start: 3, End: 7, by LightBWK (blend-file).

Edge Types

Edge types are basic algorithms for the selection of lines from geometry. When using the parameter editor you have to choose at least one edge type in order to get a render output, but several edge types can be combined in one line set. Edge types can also be excluded from calculation by pressing the *X* next to them.



Examples of some basic edge types: Silhouette (green), Crease (black), Border (blue) and Edge Marks (red) (blend-file by LightBWK).

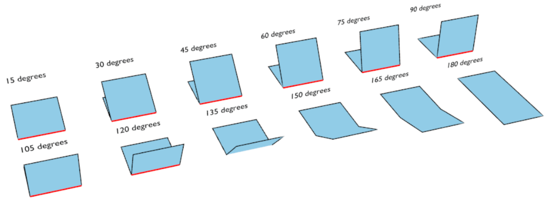
Type

Silhouette

Draws silhouettes around your closed objects by rendering lines where the surface normal transitions between pointing toward and away from the camera. It is often good for organic objects (like Suzanne & Sphere), and bad for sharp edges, like a box. It cannot render open mesh objects like open cylinders and flat planes.

Crease

Shows only edges whose adjacent faces form an angle sharper than the defined view map's *Crease Angle*.


15 degrees
30 degrees
45 degrees
60 degrees
75 degrees
90 degrees
105 degrees
120 degrees
135 degrees
150 degrees
165 degrees
180 degrees

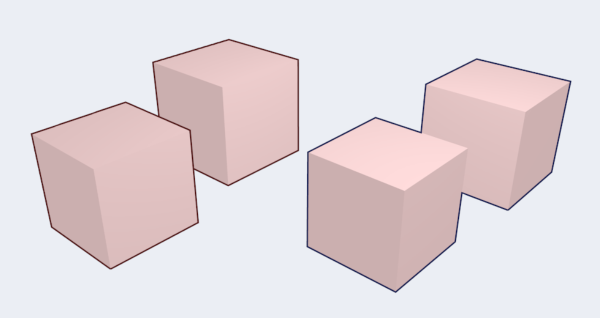
Crease Angle proof of concept for 121° by LightBWK (blend-file).

Border
: Border shows open mesh edges, i.e. edges that belong to only one face. An open cylinder has open edges at the top and bottom, and a plane is open all around. Suzanne's eye socket is an open edge.

Edge Mark
: Renders marked edges. See Edge Marks for details.

Contour
: Draws lines around each object, separating it from other objects behind it, or the scene background.

External Contour
: Draws lines around all objects, separating them from the scene background, but not each other.



Left pair: Contour; Right pair: External Contour.

Material Boundary
: Draws lines where two materials meet on the same object.

Suggestive Contour
: Draws some lines which would form the *Silhouette* of the mesh if the view point was shifted. Depends on your view map settings for *Kr Derivative Epsilon* and *Sphere Radius* (further information: File:Manual-2.6-Render-Freestyle-PrincetownLinestyle.pdf).

Ridge & Valley

Draws lines marking crests of ridges and valleys, i.e. places where the surface curvature is at its minimum or maximum. Depends on your *Sphere Radius* view map settings.

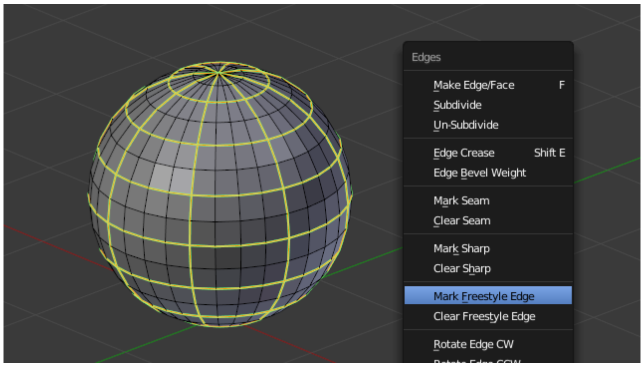
Edge Marks

In Edit Mode you can mark “Freestyle Edges” in the same manner you can mark “Seams” for UV unwrapping or “Sharp” for edge split. These marked edges are available to render when you select *Edge Mark*.

This is done as follows:

1. Select the mesh object and enter *Edit Mode*.
2. Select the edges you want to be marked.
3. Press `Ctrl-E` and select *Mark Freestyle Edge*.

Edge marks are useful when you want to draw lines along particular mesh edges. The examples below explain the use of edge marks.


Edges
Make Edge/Face F
Subdivide
Un-Subdivide
Edge Crease Shift E
Edge Bevel Weight
Mark Seam
Clear Seam
Mark Sharp
Clear Sharp
Mark Freestyle Edge
Clear Freestyle Edge
Rotate Edge CW

Marking Freestyle Edges in Edit Mode; the edge marks are highlighted in green.

With *Edge Marks* enabled, the previously-marked lines are always rendered. You can see the black contour lines and the blue lines that are made with edge marks.